PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 19/00, 9/44 — A1

(11) International Publication Number: WO 96/22575

(43) International Publication Date: 25 July 1996 (25.07.96)

(21) International Application Number: PCT/US96/00883

(22) International Filing Date: 17 January 1996 (17.01.96)

(30) Priority Data:
08/373,688 17 January 1995 (17.01.95) US
08/373,992 17 January 1995 (17.01.95) US

(60) Parent Applications or Grants
(63) Related by Continuation
US 08/373,688 (CIP)
Filed on 17 January 1995 (17.01.95)
US 08/373,992 (CIP)
Filed on 17 January 1995 (17.01.95)

(71) Applicant *(for all designated States except US)*: INTERTECH VENTURES, LTD. [US/US]; 70 Walnut Street, Wellesley, MA 02181 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: THALHAMMER-REYERO, Cristina [AT/US]; 398 Central Street, Framingham, MA 01701 (US).

(74) Agents: HOUSTON, J., Grant et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

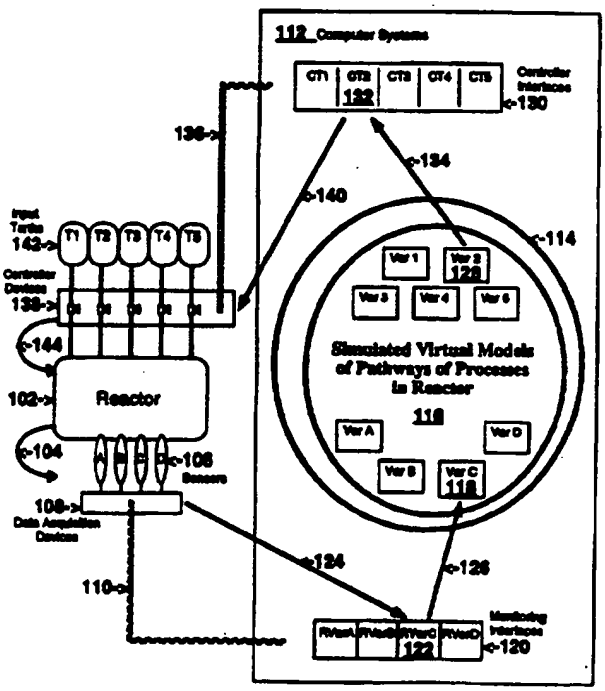
(81) Designated States: CA, JP, US, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: CONTROL SYSTEMS BASED ON SIMULATED VIRTUAL MODELS

(57) Abstract

This invention describes a computer-based system (112), methods and visual interfaces for providing an integrated development and deployment framework for visual modeling and dynamic simulation of virtual models of complex systems, which can be further integrated with monitoring (108) and control (138) devices to monitor and control the operation of the complex systems modeled (102) and can be used for information retrieval. More particularly, the virtual models in the present invention relate to visual models of biochemical complex systems, comprising sets of icons representing processes and their participants linked into multidimensional pathways (116), further organized in a hierarchy of icons representing discrete time and space compartments, wherein such compartments may contain other compartments, and wherein those modular icons encapsulate in different layers all the information, data, and mathematical models that characterize and define each virtual model.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia
AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

CONTROL SYSTEMS BASED ON SIMULATED VIRTUAL MODELS

I. TECHNICAL FIELD

A. The present invention in its broadest form relates to computer-based systems, methods and visual interfaces for providing an integrated development and deployment framework for visual modeling and dynamic simulation of virtual models of complex systems, which can be further integrated with monitoring and control devices to control the operation of the complex systems modeled and can be used for information retrieval.

Copyright Notice
A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

II. BACKGROUND ART

A. Knowledge-Based and Model-Based System for Monitoring and Control

1. Process industries, including the pharmaceutical, biotechnology, chemical, food, environmental and any other can save millions of dollars by using artificial intelligence for process optimization to control complex productions facilities. Using large-scale cultivation of microorganisms or mammalian cells are extreme cases in terms of complexity, when considering then as the individual manufacturing plants involved in complex chemical synthesis. Current systems monitor very general types of phenomena, such as gas pressure, pH, and in some occasions, the concentration of some product that correlates with cell growth or production, but those parameters are usually poor indicators of how much of the desired product is produced. . Complex mixtures of chemical reactions can be finely controlled externally by modifying the types and amounts of inputs added, if one could predict what will happen by adding those inputs, which requires a good knowledge and a model of such system of reactions. This is particularly the case with biological cellular systems that have a very sophisticated methods to transduce the signals provided by ligands in their external environment to the interior of the cell, resulting in the execution of specific functions. Such detailed and accessible mechanistic models of those pathways of reactions are not currently used for monitoring and control systems, but would be highly desirable.

2. Several **knowledge-based systems** for monitoring and control functions have been used to include the knowledge of experts into the automated control of production facilities. A knowledge-based system interprets data using diverse forms of knowledge added to the system by a human domain expert including: a) shallow knowledge or heuristics, such as human experience and interpretations or rules-of thumb; and b) deep knowledge about the system behavior and interactions. The systems that mainly based in the first type of knowledge are in general referred to as knowledge-based expert systems, and the logic is represented in the form of production rules. In the more advanced real-time expert systems, inferencing techniques are usually data-driven using forward chaining, but can also employ backward chaining for goal-driven tasks and for gathering data. The inference engine searches for and executes relevant rules, which are separate from the inference engine and therefore, the representation is intrinsically declarative.

3. **Object-oriented expert systems** allow a powerful knowledge representation of physical entities and conceptual entities. In those systems, data and behavior may be unified in the class hierarchy. Each class has a template that defines each of the attributes characteristic of that class and distinguish it from another types of objects. Manipulation and retrieval of the values of the data structures may be performed through methods attached to a object's class. **Model-based systems** can be derived from empirical models based on regression of data or from first-principle relationships between the variables. When sufficient information to model a process -or part of it- is available, a more precise and compact system can be built.

4. There is a number of **commercially available shells** and toolkits that facilitate the development and deployment of domain-specific knowledge-based applications. Of those, real-time expert-system shells offer capabilities for reasoning on the behavior of data over time. Each of the real-time object-oriented shells from various vendors offers its set of advantages, and each follows a different approach, such as compiled versus interpreted, and offers a different level of graphic sophistication. The specific shell currently selected for the implementation of this invention is Gensym Corporation's G2 Version 3.0 system, and in part Version 4.0, which is designed for complex and large on-line applications where large number of variables can be monitored concurrently. It is able to reason about time, to execute both time-triggered and event-triggered actions and invocations, to combine heuristic and procedural reasoning, dynamic simulation, user interface, database interface capabilities, and other facilities that allow the knowledge engineer to concentrate on the representation and incorporation of domain-specific knowledge to create domain-specific applications. G2 provides a built-in inference engine, a simulator, prebuilt libraries of functions and actions, developer and user-interfaces, and the management of their seamless interrelations. A built-in inspect facility permits users to search for, locate, and edit various types of knowledge. Among G2's Inference Engine capabilities are: a) data structures are tagged with time-stamp and validity intervals which are considered in all inferences and calculations, taking care of truth maintenance; and b) intrinsic to G2's tasks are managed by the real-time scheduler. Task prioritization, asynchronous concurrent operations, and real-time task scheduling are therefore automatically provided by this shell. G2 also provides a graphic user interface builder, which may be used to create graphic user interfaces which are language independent and allow to display information using colors, pictures and animation. Dynamic meters, graphs, and charts can be defined for interactive follow-up of the simulation. It also has debugger, inspect and describe facilities. The knowledge bases can be saved as separated modules as ASCII files. The graphic views can be shared with networked remote CPUs or terminals equipped with X Windows server software.

B. **Computer-Aided Physiological and Molecular Modeling and Artificial Intelligence in Molecular Biology**

1. Most computer-aided physiological and molecular modeling approaches have resulted in computer models of physiological function that are numerical mathematical models that relate the physiological variables using empirically determined parameters. Those models, which can become quite complex, aim at modeling the overall system.

2. Both molecular biology and medicine have been fields of previous activity in the application of artificial intelligence (AI). In molecular biology, although there were some early systems such as Molgen and Dendral, the activity has intensified recently as a consequence of the explosion in new technologies and the derived data, mostly related with the Human Genome project and the handling of large amounts of sequence data generated, relating to both DNA and proteins. There has also been an increased interest in computer methodologies in 3D structural models of molecular interactions. As an example, the topics covered in symposia such as the recent Second International Conference on Intelligent Systems for Molecular Biology, 1994, Stanford University, CA (see the reference to its Proceedings for a current state of the art, which are here included by reference) cover areas as wide as machine learning, automatic generation of representations, inductive and deductive reasoning, case-based reasoning, computational linguistics applied to both text and DNA or protein sequences, constraint propagation, bayesian inference, neural networks, qualitative modeling, expert systems, object-oriented databases, simulation, knowledge acquisition technologies, and knowledge base maintenance. In those and many other published papers, there are as many objectives and as many approaches as there are teams. The projects, however, are in most cases of academic interest, and in few cases it has been attempted to develop finish products for commercial distribution. Here, only two projects will be mentioned that have some common objectives with the system that is the object of this invention. Discussions over other previous approaches are also included in those references.

3. The Molgen group at the Stanford University's Knowledge Systems Laboratory has been studying for several years scientific theory formation in the domain of molecular biology, as reported in by Karp, P.D. and Friedland, P. (included here by reference). This project relates to the system object of this invention in that both "are concerned with biochemical systems containing populations of interacting molecules ... in which the form of

knowledge available ... varies widely in precision from quantitative to qualitative", as those authors write. However, the domain of Molgen is a small subset of the domain of the system object of this invention. Within the Molgen group, the Ph.D. dissertation of P. D. Karp (included here by reference), focused on applying that research to the design of a computer-assisted reasoning and hypothesis formation system. In the process, the author "developed a qualitative biochemistry for representing theories of molecular biology", as he summarizes in an abstract with the same name the AI Magazine, Winter 1990, pp 9-10. In particular, he developed three representation models to deal with the biochemical pathways related to the tryptophan operon in bacteria, called declarative device models, each having different capabilities and using different qualitative reasoning approaches. Only chapter 3, dealing with those models, have relevance in the context of this invention. The first model uses IntelliCorp's KEE (a commercial product) frames to describe biological objects and KEE rules to describe chemical reactions between the objects, which he recognizes to have serious limitations because is not able to represent much of the knowledge available to biologists. The objective of the second model, which is an extension of the fixed state-variable network of deKleer and Brown, is to predict reaction rates in a given reaction network, incorporating a combination of quantitative and qualitative reasoning about state-variables and their interdependencies. The drawbacks are that this model is not able to incorporate a description of the biological objects that participate in the reactions, and in addition it does not have a temporal reasoning capabilities, representing just an static description of the state variables and their relationships. The third model, called GENSIM and used for both prediction and hypothesis formation, is an extension of model 1, and is composed of three knowledge bases or taxonomical hierarchies of classes of a) biological objects that participate in the trp-operon gene-regulation system, b) descriptions of the biological reactions that can occur between those objects, and c) experiments with instances of those classes of objects. The GENSIM program predicts experimental outcomes by determining which reactions occur between the objects in one experiment, that create new objects that cause new reactions. GENSIM is also further used in conjunction with the HYPGENE hypothesis-formation program that will not be further discussed here. Characteristics of the GENSIM program that may be relevant, although different, for the system of this invention are:

a) chemical objects are homogeneous populations of molecules, objects can be decomposed into their component parts, and identical objects synthesized during a simulation are merged;
b) chemical processes are frames arranged in an inheritance class hierarchy, which represent reactions between those populations as probabilistic events with two subpopulations, one that participate in the reaction and one that does not;
c) restrictions are specified in the form of preconditions for chemical reactions to happen.
d) those Forbus-style processes can create objects and manipulate their properties, but cannot reason about quantitative state variables such as Quantities. In his words, "processes are described as KEE units... [that] .. specify actions that will be taken if certain conditions hold", and in that sense are like production-rules;
e) temporal reasoning is not available, resulting again in static representations and simulating only behavior in very short time intervals.

4. This system integrates a variety of forms of knowledge representation, some of them totally novel, while some of these forms may have been given a previous treatment by other authors which may be similar in some of its aspects to the treatment given in this invention. However, upon integration into a totally new approach, that treatment becomes a part of a novel representation and innovative system. For example, regarding the semi-quantitative simulation component, which is just one of several applications offered by the system object of this invention. L.E. Widman (1991) describes a semi-quantitative simulation of dynamic systems in a totally different domain. What both systems have in common is the assumption that, as stated by Widman: "a semi-quantitative model ... is intended to predict correctly the direction in which each variable will change and to provide a correct partial ordering of the magnitude of change relative to other variables in the model. ... The questions can be answered in terms of relative quantities rather than absolute quantities. ... model parameters that are not specified explicitly are given the implicit, default values of 'normal' (unity) ... ". Similar but different concepts are applied in developing part of the system of this invention, but as it will become clear from the detail descriptions in the following sections, not only are they applied to a new domain, but the innovative tools and methods used in the present implementation are also quite different. For example, while he assumes that " ... the default, or implicit, value of 'normal' maps onto unity for parameters and onto zero for variables...." the assumption in the prebuilt modular components in the current implementation differs from that statement in that the default, or implicit, value of 'normal' may map onto values other than unity for parameters and onto values other than zero for variables, with those values being defined based on expert knowledge.

III. DISCLOSURE OF INVENTION

A. This invention describes an integrated computer-based system, methods and visual interfaces for providing a development and deployment framework for visual modeling and dynamic simulation of virtual models of

complex systems, which can be further integrated with monitoring and control devices to monitor and control the operation of the complex systems modeled, based on the real-time simulation of those Virtual Models. The system of this invention can be used by scientists to build the Virtual Models, which in turn are to be used by process engineers to design, monitor and control a variety of processing units. The Virtual Models can also be used for information retrieval, by using the set of visual interfaces provided to perform a variety of tasks. The available information and data about those complex models is stored into modular, modifiable, expandable and reusable knowledge structures, with several layers of encapsulation which allows to hide or display the details at the desired level of complexity. More particularly, the virtual models in the present invention describe visual models of biochemical complex systems, comprising sets of icons representing processes and their participants linked into multidimensional pathways, further organized in a hierarchy of icons representing discrete time and space compartments, wherein such compartments may contain other compartments, and wherein those modular icons encapsulate in different layers all the information, data, and mathematical models that characterize and define each virtual model.

B. Process industries can save millions of dollars by using different methods of process optimization to control production facilities. Using large-scale cultivation of microorganisms or mammalian cells are extreme cases in terms of complexity, when considering then as the individual manufacturing plants involved in complex chemical synthesis. Microorganisms are the preferable systems for producing natural substances that have a multitude of uses, such as drugs, foods, additives, biodetergents, biopolymers, and other new and raw materials. Most particularly, mammalian cells are the preferable systems for producing potent active substances for therapeutic and diagnostic uses. The ultimate level of complexity is using a whole animal as the live factory for continuous production for important secreted proteins. However, the current systems only monitor very general types of phenomena, such as gas pressure, pH, and in some occasions, the concentration of some product that correlates with cell growth or production. For example, for controlling the production of a particular secreted protein that may be produced in very low amounts in relation to other protein, the total protein amount of protein is measured, which is a very poor indicator of how much of the desired protein is produced. Complex mixtures of chemical reactions can be finely controlled externally by modifying the types and amounts of inputs added, if one could predict what will happen by adding those inputs, which requires a good knowledge and a model of such system of reactions. This is particularly the case with biological cellular systems that have a very sophisticated methods to transduce the signals provided by ligands in their external environment to the interior of the cell, resulting in the execution of specific functions. Such detailed and accessible mechanistic models of those pathways of reactions are not currently used for monitoring and control systems, but would be highly desirable. This invention provides the system and methods that allow to: a) build detailed mechanistic models of the complex systems involved, and b) to use inference methods that integrate the simulation of those Virtual Models with inputs from monitoring devices to allow for the intelligent control of the operation of the complex system.

C. The accuracy and validity of knowledge-based systems correlates not only with the quality of the knowledge available to the developers but also with their ability to understand, interpret and represent that knowledge. Because of the complex interrelationships driving the biochemical processes within and between cells, it is necessary to provide the many options for knowledge representation required by those systems. This invention presents a hybrid dynamic expert system that combines: a) the inheritance and encapsulation features characteristic of an object-oriented modeling approach, b) the procedural and rule-based inference capability of an expert system, and c) a model-based simulation capability. It is a further advantage of this invention to provide a framework that: a) allows domain-experts to directly enter their knowledge -to create and modify visual models, as needed based on additional experimentation-, without the need of knowledge engineers as intermediaries, and b) provides a knowledge-base having a well defined structure for representing knowledge about entities, processes, pathways and interacting networks of pathways, providing a visual interface and associated methods capable of dealing with incomplete and constantly evolving information and data. The data structures and domain-specific knowledge-base are independent of application-specific use, allowing the application-specific knowledge-bases to expand without affecting the basic operation of the system. Specific applications can be quickly built by using the prebuilt building blocks and the paradigm of "Clone, Connect, Configure, and Initialize". The structure of the domain-specific knowledge-base serves also as the infrastructure provided to store additional information about new objects and models. The innovations of this invention include but are not limited to the specific design, generation, integration, and use of the libraries of building-blocks, access panels, and their associated methods, that allow representation, interpretation, modeling and simulation of different types of entities and their states, relations, interactions, the pools of each of those entities in different compartments, the processes in which those pools of entities participate, their discrete compartmentalization in time and space, as well as the concepts that make the simulation of this very complex systems possible. A large Virtual Model can be built as a set of modules, focusing each on different subsystems

Each of the modules can be run on top of the repository module that contain the class definitions and associated methods, and they can be dependent or independently of each other, or they can be maintained in separate CPUs and seamless integrated by the Shell's supervisor.

D. This invention includes libraries providing the prebuilt building-blocks necessary for visually representing the vast breath of knowledge required for building virtual models of complex systems in general, and of biological systems in particular. The building-blocks are classified in palettes that can be selected through system menus. The descriptive information and data and the mathematical models are all encapsulated within the modular components, in the form of attribute tables and in the form of component icons, with a plurality of methods are associated with each of the icons. The visual models are built by interconnecting components of each reservoir icon to components of the one or several process icons in which their entities participate as inputs or outputs, or by interconnecting components of each process icon to components of the one or several reservoir icons that provide inputs or receive outputs to that process icon, resulting in complex networks of multidimensional pathways of alternating layers of reservoir icons and process icons, which may be located in the same on in different compartments. The time compartment icons, the reservoir icons, and the process icons or their components comprise sets of quantitative variables and parameters, and a set of associated methods that permit real-time simulations of the models created with those modular components. The system of this invention combines a number of programming paradigms, integrating the visual interface to define and constrained objects represented by icons, with procedural and inferential control and quantitative and semi-quantitative simulation methods, providing a variety of alternatives to deal with complex dynamic systems.

E. Of importance in simulating the behavior of complex systems is the need to model the different quantities and states at which the entities can be found in particular locations at different points in time, and also to model the events that cause the transitions from one state to another, or the translocation from one location to another, or the progress to the next phase in the time sequence. Some of the important teachings of this invention comprise: the representation of those states, transitions, and locations; the methods to implement their graphic modeling; and the methods to dynamically simulate the dynamics of the pools of entities in each state, location, or phase. In the particular domain implemented to illustrate this invention, there are several major types of states and transitions to be considered, depending of whether the entity to be considered is a biological system, organ, cell, cellular compartment, molecule or any other of their components. We are providing here with just a few examples considered in the currently preferred embodiment of this invention.

F. The architecture of the system of this invention allows various bioProcesses to simultaneously compete for the contents of a bioPool while also allows one bioPool to participate in various bioProcesses in different capacities, with different units within a bioPool behaving in different ways, as determined by the class of bioReactant to which they are distantly connected via the bioPosts. With this implementation, all bioEngines that input units into a given bioPool and all the bioEngines that get input units from that bioPool are integrated. At the same time, all bioPools that input units into a given bioEngine and all the bioPools that get input units from that bioEngine are also linked. The result is a very complex multidimensional network of composite bioObjects, which provides the matrix for further reasoning and simulation by the program. For example, the formulas that provide the values for the encapsulated sets of variables and parameters refer to connected bioObjects, both graphically or distantly, providing the capability of concurrently and dynamically compute as a large number of processors arranged both in parallel and in series within that matrix.

G. The modeled system's behavior is defined by mathematical components, represented by a set of model differential and algebraic equations that provide the values of the system's variables and describe their behavior, together with the set of associated parameters that control the behavior of the variables and the system as a whole. The system's variables and parameters are embedded and distributed throughout the system of connected structures, encapsulated within the subcomponents that define the system's architecture. The model can then be viewed as a set of embedded block diagram representations of the underlaying equations that can be used for dynamic numerical simulation and prediction of the effects of perturbations on the system, and to ask what-if type questions.

H. The compartmentalized bioModels, and the methods attached to them, encode knowledge that enables the program to reason about the containment of different parts of the model in several compartments, while the architecture of the network of linked bioObjects of diverse types is transparently maintained, regardless of the transfer of the bioObject icons to different locations. They also comprise quantitative variables and parameters, some relating to quantities and rates translocation, distributed either within the corresponding reservoirs and process or relating to time intervals which may be associated with the time compartments, all of which have associated simulation formulas to compute their values. In addition, the modeler can define expert rules to monitor the values of those variables while the simulation is running, and to either set other values or control the course of the simulation in a variety of ways. The expert rules can also reason about time or about

events resulting from the simulation, Inference using those types of knowledge may direct further actions to be executed by the system.

I. The visual interface further provides quick access to several automated methods for compiling, retrieving, and displaying the modular components of the visual models as well as the information and data they contain. The system integrates inferential control with quantitative and semi-quantitative simulation methods, and provides a variety of alternatives to deal with complex dynamic systems and with incomplete and constantly evolving information and data. A number of functions and visual interfaces can be selected from the menus associated with each of those icons or their components, to extract in various forms the information contained in the models build with those building blocks:

1. A function is provided by the methods to create and display of interactive networks of pathways, by programmatically integrating the components of the Virtual Model into multidimensional networks of interacting pathways, including branching and merging of pathways, cross-talk between pathways that share elements, and feed-back and forward loops;
2. A function is provided by the methods to perform complex predefined queries that combine criteria related to the structural composition of the bioEntities involved, the position of bioPools downstream or upstream of the bioPool taken as reference, the role that those pool of entities play in processes, the location of those pools and processes in the discrete time and space compartments, or any combination of them.
3. A function is provided by the methods to dynamically simulate their continuous interactions, modifications and translocations to other compartments, and the time-dependency of such interactions, optionally using the encapsulated absolute-valued or scaled-valued parameters and variables.

J. The examples provided to document the current implementation of this invention focus on modeling biochemical regulatory processes that are relevant for intracellular or intercellular signaling. This is however one of the many potential applications of the system of this invention, and it should not be construed to limit the applicability of this system to the numerous other applications that involve complex systems in any other domain, which can be developed by minor modifications or additions of the system using the methodology here presented.

K. There are numerous other uses of the core system, methods and visual interface of this invention, in addition to the monitoring and control applications described here. Of particular interest is the modeling and simulation of disease specific conditions and the testing the effects -both desired and unwanted side effects- of potential therapeutic agents. This invention also allows to analyze disturbances such as potential environmental or biological inducers of disease in both physiological and pathophysiological models. The methods included in this invention can in a similar way be used in a variety of applications, including but not limited to: simulation and prediction of experimental results; interactive drug design; study of drug side-effects and multiple drug interactions; simulation of interactive causes in the induction and progression of disease, including both biological and environmental factors; diagnostics and clinical decision support; therapy planning; theoretical research; and numerous other applications.

L. The foregoing and additional objectives, descriptions, features, operations and advantages of the present invention will be understood from the following detailed description of the preferred embodiments in combination with the accompanying figures.

IV. BRIEF DESCRIPTION OF DRAWINGS

FIG.1 is a high-level illustration of the various components integrated in the system of this invention to use the virtual models for control functions.

FIG.2 is an schematic representation of the organization in the system of this invention of the components of the virtual models in the domain of cell biology.

FIG.3 is an schematic representation of the organization in of domain-specific processes in discrete space compartments an time compartments.

FIG.4 is an schematic representation of the handling of the dynamics of the progression of populations of cells through different states by means of the sets of pools of cells and processes characteristic of this invention.

FIG.5 is an schematic representation of the multiple layers of linked pools of entities and processes the result in the multidimensional pathways characteristic of this invention.

FIG. 6 is a more detailed representation of how the various iconic components encapsulated in the icons representing the pools of entities and processes are linked in the current implementation of this invention.

FIG.7 is a detailed representation focusing on the iconic components of a process and their relations to its sources of inputs and the targets for its outputs.

FIG.8 is a detailed representation focusing on the iconic components of a pool of entities and its inputs and outputs.

FIG.9 shows domain menus that allow different types of access to different types of capabilities, for the purpose of modeling or using the virtual models, here focusing on the developer mode that allows developing additional tools.

FIG.10 is the table of attributes of a complex variable structure that allows the integration of measured values and the simulated values that facilitates the implementation of this invention.

FIG.11 is a library of predefined classes of variables with associated generic simulation formulas associated with each class available to the modeler virtual models for a specific domain.

FIG.12 describes the sets of attributes of the components of a reservoir, focusing on the variables and parameters of its model-block component..

FIG.13 describes the sets of attributes of the components of a pool of entities, focusing on its variables and parameters.

FIG.14 describes the sets of attributes of the components of a process, focusing on the variables and parameters of its reactants.

FIG.15 describes the sets of attributes of the components of a process, focusing on the variables and parameters of its engine and products.

FIG.16 shows domain menus of the modeler mode that allow modelers to access the different palettes of prebuilt building-blocks.

FIG.17 shows examples of different palettes of prebuilt molecular components.

FIG.18 shows an example of a complex molecular structure built with the prebuilt molecular components.

FIG.19 shows a palette with examples of prebuilt model-blocks.

FIG.20 shows a palette with examples of prebuilt composite processes.

FIG.21 shows a palette with examples of prebuilt reactants.

FIG.22 describes the tools for interactively establishing links between components that result in multidimensional pathways.

FIG.23 shows a palette with examples of prebuilt composite domain-specific compartments..

FIG.24 describes a domain-specific compartmentalization of the components of a virtual model, in this case focusing on the sequential phases of a cell's cycle and on its subcellular compartments.

FIG.25 describes a domain-specific implementation of compartmentalized model interacting with each other and with the external environment.

FIG.26 describes a domain-specific characterization of the compartmentalized model from an external point of view.

FIG.27 describes a domain-specific implementation of an evolutionary tree representing the alternative successive states that a compartmentalized model may follow depending on the events that depend on their internal processes responding to the environment, from an external point of view.

FIG.28 is an schematic representation of the combination of the inference and simulation capabilities used in this invention to simulate virtual models of complex systems.

FIG.29 is a detailed representation of how in this invention pools of cells interact with pools of molecules as reactants of processes which products are either molecules or cells in a different state.

FIG.30 describes an example of the predefined Query Panels and their use.

FIG.31 describes how a user can request from any process within the virtual model the dynamic generation of the pathways of all the processes that are either upstream or downstream fro m that process.

FIG.32 describes an example of the predefined Navigation Panels that the user can request from any reservoir within the virtual model for the dynamic generation of constrained pathways of all the processes that are either upstream or downstream fro m that reservoir, but which are contained within a compartment selected by the user.

FIG.33 describes an example of the predefined Simulation Panels that the user can request from any reservoir within the virtual model the dynamic generation of constrained pathways and for the control of the dynamic simulation of the kinetics of those pathways.

FIG.34 describes how such simulation can be followed by plotting the time-series of any of the quantitative values of any reservoir and process in the subsystem being simulated.

FIG.35 describes an example of the predefined Experiment Panels that the user can select from the domain menus to request the dynamic generation of constrained pathways from multiple initial points and for the control of the dynamic simulation of the kinetics of those pathways.

FIG.36 describes an example of such set of pathways from multiple initial points

V. BEST MODE FOR CARRYING OUT THE INVENTION

Notes

1. The body of the present application has sections that may contain some discussion of prior art teachings intermingled with discussion of innovative and specific discussion of the best mode to use that prior art in this invention as presently contemplated. To describe the preferred embodiments, it is necessary to include in the discussion the capabilities offered by the shell used as development and deployment framework for this invention (hereafter referred to as "the Shell"). The applicant specifically notes that statements made in any of those sections do not necessarily delimit the various inventions claimed in the present application, but rather are included to facilitate the process to explaining how the workings of an existing set of tools is used to illustrate the preferred embodiments of the new tools and applications claimed in the Claims section. The currently preferred embodiment of this invention, as described in the present application, is based on the definitions of a particular Shell: Gensym Corp.'s G2 Expert System. There are several other attributes that relate to the Shell's built-in performance and formatting capabilities, which are not shown in those examples. Some information included within the body of this application was extracted from various sources describing the characteristics of G2, including user manuals, and some of this material is subject to copyright protection.

A. Monitoring and Controlling the Operation of a Reactor Using a Simulated Virtual Model

1. The system object of this invention is a hybrid combination of model-based methods describing the explicit mechanistic reference behavior of the production system with the input from process sensors that monitor the state of the production system, triggering event-based control flow of operations characteristic of each system. The system implements rules that wait for events to happen. Such events may be complex combinations of individual events, such as selected measured values reaching certain predefined or simulated values, or be within certain predefined ranges, including the implementation of fuzzy-logic within those rules. Such rules may fired other rules or invoke inferencing procedures or cause certain control actions to take place. This system can detect the current status of the production system, and when the selected monitored variables reach certain values specified by the simulated values of those variables, then certain control action take place. This system incorporates monitoring capabilities with the knowledge-based model to provide expert control of the operation of the production system or biofermentor. This control system provides simultaneous supervision of numerous operating variables (such as intermediary or end-products, which in the case of cellular systems can be intracellular or secreted), and compares them with the simulated values of those variables resulting from the encapsulated mathematical models. Depending on the dynamically monitored behavior of the production system, the control system is able to compensate by feeding components at different rates, feeding additional components, or stop feeding some of the previous components.

2. Because of the user friendly and intuitive interface, the bench scientists can participate in the design of the production system by editing the knowledge base and the visual models themselves, incorporating information and data obtained in their own experimental research or that from other published resources. It allows the direct incorporation of research into the scale-up operations. The scientists can rapidly build models by using the paradigm Clone, Connect, Configure, Initialize. They select the desired building block from the many palettes provided, clone the building block and drop it on desired compartment, connect the building block to other building blocks in the model, configure by entering values of attributes (optional), and initialize the building block to establish relationships with other building blocks in the model. The information and data is entered into the system as attributes of objects, either as user inputs, or directly from measuring devices using and interface between said measuring devices and the computer system.

3. The system provides a graphical computation and control language, where the objects communicate through the links established between them. Some of those linkages are built in within the composite prebuilt building blocks. The scientist building the model clones said prebuilt building blocks, as needed for each part of the model, and establishes the links between appropriate components within different building blocks, as needed. Other information, such as the name, description, references, or the values of parameters specific for each component are entered by the modeler in the attribute table associated with each icon. In addition, an attribute may be a variable.

4. Variables in this system are themselves objects, and maybe of two major classes: a) one-valued variables have only one value which may be either provided during a simulation by a simulation formula or a simulation procedure, or can be inferred by any other means, such as rules or general formulas defined by the modeler; or b) two-valued variables have two values: the simulated-value may as before be either provided during a simulation by a simulation formula or a simulation procedure, or can be inferred by any other means, such as rules or general formulas defined by the modeler, and the measured-value is provided in real time through an external sensor mapped to said variable. The one-valued variables are used in such components by default

because of their smaller footprint, and their values are provided by default by generic simulation formulas that are specific for each class of variable. However, the modeler can replace them with the equivalent subclasses of two-valued variables for each instance of a component where the variable is mapped to an external sensor, and when both the measured valued and the simulated value of such variable is desired. The modeler can also write specific formulas for any desired instance of a variable, which then overrides the default generic formulas. It is possible, through any of the inference mechanisms to compare the measured value and the simulated-value, (either the current values or the mapped at some time point in the past, since the system is able to keep a time-stamped history of both types of values), and take specified actions when the inference criteria are met. Such actions include, but are not limited to: cause for a valve for a component feed to be more or less open or closed, activation or deactivation or whole branches of the model pathways being simulated. Such actions can alternatively be invoked as a result of comparing either the measured-value or the simulated-value to any predefined constant value or range of values, or the value of any other variable or parameter in the system. Or the action or set of actions could result from inferences involving any number of comparisons between measured-values or simulated-values for any number of variables, or any number of parameters, or any predefined constant values.

5. As shown in FIG.1, the system system of this invention is composed of:

a) a reactor (102) which may comprise any combination of reaction tanks, fermentors, bioreactors, or other processing units;

b) one or more data acquisition devices (108) which may comprise any combination of hardware and software devices, such as sets of sensors (106) that measure the amounts of selected chemicals in the reactor, signal transducers, filters, amplifiers, data acquisition boards, appropriate device drivers, or any other required devices;

c) one or more computer systems (112) comprising CPU, memory, storage device, display, user input device, linked (110) to the data acquisition devices (108);

d) one or more computer programs (114) and one or more computer Virtual Models (116) of pathways of chemical interactions and other processes in the reactor (102), wherein the computer programs (114) and the Virtual Models (116) are used to quantitatively or semi-quantitatively simulate in real-time the Virtual Models (116), wherein the Virtual Models encapsulate one or more variables (118) that represent the quantities of certain entities in the reactor(102) and one or more variables (128) that represent the quantities of certain entities or that represent certain events which when reaching certain values during the simulations of the Virtual Models (116) by the computer programs (114) trigger certain control actions that affect the operation of the reactor (102), and wherein both the computer programs (114) and the Virtual Models (116) are loaded in the memory of one or more of the computer systems (112);

e) one or more monitoring interfaces (120) loaded in the memory of one or more of the computer systems (112), which act as bridges or software interfaces between the data acquisition devices (108) and the computer programs (112) and allow passing of values (124, 126), such as the values of the amounts of certain entities in the reactor (102), as measured by the corresponding sensors (104), to the corresponding variables (118) embedded in the Virtual Models (116) that represent said amounts;

f) one or more controller devices (138) which regulate the operation of the reactor (102), such as controlling the flow of certain inputs (142) to the reactor (102), wherein the controller devices (138) are linked (136) to the computer systems (112);

g) one or more controller interfaces (130) loaded in the memory of one or more of the computer systems (112), which act as bridges or software interfaces between the computer programs (114) and the controller devices (138) and allow passing of control signals generated by the computer programs (114) as a result of the values of any combination of any number of variables (128) embedded in the Virtual Models (116) reaching certain values.

6. FIG.1 also shows the directions of flow of data and control in the system of this invention. The amounts of certain entities, which are specific for each particular process design, are captured (104) by the corresponding set of biosensors (106) and, through the data acquisition devices (104), those values are passed (124) to the monitoring interface variables, which in turn pass those values (126) to the corresponding variables (118) in the Virtual Models (116) that represent those values. Those variables (118) are integrated with many other variables and parameters embedded in the Virtual Models (116) during the real-time simulation of the Virtual Models (116), including other variables (128) embedded in the Virtual Models (116) that represent quantities, rates, or other events, and which are monitored by the programs (114) during the simulation. Whenever during the simulation the values of any combination of any number of said monitored simulated variables (128), which are specific for each particular process design, reach certain values, then the programs (114) pass control signals (134), through the appropriate control auxiliary structures (132) in the controller interfaces (130), which are forwarded (140) to the controller devices (138), which in turn control the flow of inputs (142) and regulate (144) the operation of the bioreactor (102), which is being monitored (104) in a continuous manner

7. Depending on the application requirements, the interfaces may provide bridges to Supervisory Control and Data Acquisition (SCADA) systems (such as the HP's RTAP SCADA or others), Distributed Control Systems (DCS) (such as Honeywell TDC3000 DCS or others) or Programmable Logic Controllers (PLCs) (such as Allen-Bradley's PLC3/PLC5 families, or others), with the adequate protocol drivers, as well as to relational databases, object-oriented databases, ASCII files, as well as to a number of other connectivity applications, allowing the program to send or receive values over said interface. The program (114) and the interfaces (120, 130) are separate processes which may be located on one or more host computers. In the later case, a communications link, such as a TCP/IP or DECnet protocols based link and port, is required. The program can also use various interfaces simultaneously, linked to different types of devices. The interface serves as a bridge between selected variables embedded in the objects of the Virtual Models and their corresponding mappings in one or more external systems. Functions defined in the interface can be invoked by remote procedure calls (RPCs) defined in the program, and viceversa.

8. The ideal system for the intelligent control of biotechnical processes must posses a large set of features reflecting the real-time aspects of the control system, as well as the specific characteristics of the biochemical processes involved. There are a variety of new sensor technologies which may be used to provide the monitoring capabilities external to the Virtual Models. Furthermore, the system of this invention allows to integrate a variety of real-time control options with the ability to use mechanistic models to drive the automatic decision making support for process control.

9. The knowledge-based Virtual Models include model-based reasoning that models the dynamic behavior of processes on the reactor. This mechanistic approach may involve any number of variables to be monitored, including those measured and monitored in the reactor and those simulated and monitored in the Virtual Models, which are both used as inputs for the inference engine to be compared against each other or against other specified values, and which meeting the specified conditions trigger actions that result in control operation on the reactor. The Virtual Models provide a visual qualitative and quantitative description of processes that happen inside of the reactor, as well as a description of the participants in those processes. In some cases, the reactor may contain chemicals in solution or bound to other structures, such as carriers or membranes. In other cases, the reactor may contain less defined mixtures resulting from cell extracts. Yet in other cases, the reactor may contain cells or one or more types, interacting with each other and with their environment, the culture medium, where each of such cells is an reactor itself that interact with other cell reactor. The system of this invention provides capabilities to produce more or less detailed Virtual Models of any of those types of reactors, or Virtual Models of subsystems within such reactors.

10. The program provides a domain-specific framework for continuous and discrete process modeling and simulation, which combines a hierarchical object-oriented representation to provide facilities for hierarchical systems modeling to handle multi-dimensional information, model-based and knowledge-based reasoning, temporal inferencing, and dynamic real-time monitoring and control. Furthermore, the system of this invention separates the representation of the physical systems as Virtual Models from the monitoring and control aspects, allowing to integrate the same Virtual Models with different combinations of monitoring and control designs, to solve different production needs.

11. The present invention refers to a domain-specific, application-independent, knowledge-based and model-based, object-oriented, iconic, real-time computer system capable of being used by scientists as a framework to construct specific and interactive information, modeling and simulation applications in the chemical and biochemical domains. The present invention also provides a variety of domain-specific, application-independent tools, graphical interface and associated methods to provide users the capability to extract, interactively or automatically, the integrated knowledge contained in those applications or models build by the modeler, and to further use those models, among other uses, to navigate through the pathways or processes and explore those processes and their participants, or for quantitative real-time dynamic simulations. The computer system comprises a plurality of methods, hereinafter called "the methods" and diverse sets of objects, some of them representing either entities or concepts, hereinafter called bioObjects, and other representing other auxiliary structures, which in general are referred as tools. Objects are arranged in object-class hierarchies and workspace hierarchies. Methods may be associated with a class of objects, an individual instance of a class, or a specified group of instances within a class. Libraries of prebuilt knowledge-based generic bioObjects are provided as the building blocks that can be combined in prescribed ways to create diverse and new knowledge structures and models.

12. In the description of this invention, the following clarifications should be noted:

a) a person using the underlaying Shell to define the domain-specific application-independent but knowledge-based classes of objects, their associated methods, and the prebuilt knowledge-based building blocks is hereinafter referred to as a "developer", while the person using those building blocks to build

application-dependent models, or expanding the libraries of prebuilt bioObjects, is hereinafter referred to as "modeler", and the person that extracts and uses the accumulated knowledge and runs simulations, is hereinafter referred to as a "user".

b) text expressions are case insensitive, and what is referred in the description as bioObject or bioProcess, may be referred to in the code as bio-object or bioprocess, or the names may appear in the code in low case or all capitalized;

c) a window refers to a display of the program on a computer screen, while a workspace refers to one of the many containers of objects that may be displayed within a window, and the subworkspace of an object is the workspace (only one) that may be associated with that object and may contain the components of that object (composite object);

d) the selection of a menu option is equivalent to clicking on that option, and both expressions are used in the following description of this invention;

e) expressions similar to "selection of option A displays B" is a short form meaning something similar to "selection of option A causes the invocation of the action or procedure specified in the definition of option A and, as a result of the execution of such action or procedure, B is displayed in the window from which option A was selected"

13. The preferred embodiment of this invention integrates a variety of knowledge representation techniques, as required for creation of virtual models of complex systems. The accuracy and validity of knowledge-based systems correlates not only with the quality of the knowledge available to the modelers but also with their ability to understand, interpret and represent that knowledge. The object-oriented approach provides a powerful knowledge representation of physical entities (such as organs, cells, DNA, enzymes, receptors, ligands, mediators, or ions) and conceptual entities (such as processes and cellular interactions, quantities and rates). In this system, data and behavior are unified in the objects. Two characteristics of object-oriented environments, encapsulation and inheritance, are very important for the design and implementation of the system of this invention. Objects are defined following class hierarchies in which the definition of each class specifies the types of attributes characteristic of all subclasses and/or instances of that class. Encapsulation permits to hide the details behind each object, and encapsulation is implemented in two different forms: a) at the attribute level, is the standard form of encapsulation of object-oriented approaches; and b) at the workspace level, a less common form of encapsulation related more to the iconic approach. Multiple levels of workspace encapsulation are supported, allowing modules with a multilayered structure with increasing levels of detail. Since the subworkspace is not inherited through the class hierarchy, neither are the components. However, once a generic instance for a class is completed with components in its subworkspace, it can be added the corresponding palette as a prebuilt building block, and the resulting composite object can be cloned, in which case all encapsulated levels of subworkspaces are also cloned. The interpreted domain-specific framework approach is easy to use and allows rapid building of iconic models. The domain-expert knowledge is represented in an easy to understand declarative from, separated from the methods that specified how that knowledge is used by the program. The iconic approach allows the user configuration for building of models to consist primarily of connecting iconic objects and filling out their tables of attributes. The visual framework enables reasoning about the interactions between objects and their compartmentalization. This graphical programming capability is particularly important because it allows the continuous expansion and modification of the system, necessary for this type of applications, by scientist modelers with no programming skills. Its modularity allows to delete, modify or create parts of the system without affecting the operation of the rest of the system. The knowledge can also be extended and reused under different contexts, and in different new applications. The same hierarchic architecture used to develop the bioObjects can be extended by the modeler to expand the libraries of bioObjects creating new objects or by cloning, configuring and modifying existing ones. With its iconic format, the system can be easily browsed and understood by the user.

14. The principle followed in the design and implementation of this invention is the breaking down of the knowledge about entities, processes and pathways down to smaller functional units, to a level where the following requirements can be met: a) allowing their repeated use as building blocks in a variety of situations, b) allowing access to the structures and processes that are susceptible to control and regulation; and c) keeping the number of units manageable. Those functional units, hereinafter called composite bioObjects, are further broken down into operational and standardized knowledge and data structures. These generic but domain-specific iconic objects add flexibility and allow the user to open the tables of individual instances to input data, modify existing bioObjects, or create new types of bioObjects. Following such principle, the system of this invention comprises a variety of domain-specific knowledge structures, all encoded as objects, which are either: a) iconic bioObjects on workspaces, b) standard graphic user interface objects such as buttons, data displays or input panels, c) non-iconic objects which are either pointed to as attributes of other objects, such as domain-specific variables, parameters, lists or arrays, or d) methods, which refer to the bioObjects or their

attributes or their components, such as actions, formulas, relations, rules and procedures, and which can be invoked interactively by the user from either the bioObjects or their components or from the standard graphic user interface objects, or invoked programmatically by other rules or procedures, and which are hereinafter referred to as the associated methods. It is an object of this invention the manner in which the combination of those types of knowledge structures encode, in a distributed form, the data and the knowledge that enables the system to compute and to reason about conditions and events defined by the developer, the modeler, any other external source, and/or generated during a dynamic simulation. Based on those conditions, the knowledge structures are used by the Shell to infer or simulate different behaviors, while data, information and/or control are propagated and actions are executed. As can be observed by comparison of the prior art previously discussed with the description in the sections to follow, the form of knowledge representation in the iconic models, libraries of composite bioObjects, graphical interface and associated methods, and the overall organization of the system object of this invention offer a very different and innovative approach that not only integrates the qualitative and quantitative description of chemical and biochemical objects with a set of state and dependent variables, such as amounts and rates, but also incorporates temporal reasoning and generates dynamic simulations.

15. Therefore, one innovation of the present invention is the iconic interface and associated methods used to partition the domain knowledge into modular and interactive knowledge structures represented as iconic objects, the connections, interactions and relations between those structures, and their behavior, incorporating declarative and procedural information. Each independent functional iconic building-block, a composite bioObject, comprises a set of iconic bioObjects and other objects and represents either:

a) the descriptive characterization of single units of molecular entities, molecular subcomponents, or molecular complexes, hereinafter called in general bioEntities;

b) the aggregation of a number of units with the same characteristics, defined as a population of such molecular entities or molecular complexes in specific states, hereinafter called a bioPool, which together with the potential inputs to and outputs from such bioPool is encapsulated in what is hereinafter called a bioReservoir;

c) the interactions between fractions of different such populations of molecular entities or molecular complexes, or their transitions from one molecular state or discrete compartment to another over time, hereinafter called bioProcesses.

d) the organization and integration of networks of pathways composed of sets of connected bioReservoirs and bioProcesses organized in compartments, hereinafter called bioModels. Such low level bioModels can be further organized in discrete location and time compartments, representing the subcellular organelles and the phases of the cell cycle respectively, which are themselves components of higher level compartments representing single cells. All those types compartments are themselves bioModel subclasses at different levels of complexity and detail in a hierarchy of subworkspaces.

16. BioObject, a subclass of the class objects provided by the Shell, is the superclass at the top of various hierarchies of classes of objects such as the class hierarchies below bioEntity, bioReservoir, bioProcess, bioModel, and others. Each instance of a class is characterized by the set of attributes defined for the class, which can be inspected in the table of attributes attached to each object by selecting it with the mouse. Some attributes define configuration information, while other attributes describe the composition and characteristics of the object, and still others hold dynamic state information, such as the current value(s), data histories, and status. The value of some attributes of an object, such as the variables and parameters, can be dynamically modified at runtime.

17. The knowledge-based modeling and simulation system of this invention interprets the information and data contained in the BioObjects' data structures, based on additional knowledge contained in the form of rules, formulas or procedures. Model-based knowledge is integrated and encapsulated in the structural, functional and behavioral models represented by the virtual iconic models or bioModels. Those virtual models are defined qualitatively by the locations, connections, and relations of their components, and quantitatively by their encapsulated mathematical models. The constants, parameters and variables that define those models are distributed through the different types of bioObjects, defined as their attributes, and the corresponding formulas and functions relate those data structures to others and characterize the particular system. The system integrates propagation of values, inference and control throughout the pathways. Model-generated results are used as input information for the inference engine, and dynamic models can also reason about the historic values of its variables as well as projecting values of variables into the future. All those different types of knowledge are added by human domain experts to incrementally build an integrated Virtual Model.

18. In the current implementation of this invention, the variables embedded in the model that are set to get values from the sensors (such as the concentrations of the corresponding bioPools) inherit their properties from two parent classes: a class of float-variable and a class of interface-data-service. One of the attributes of the later

class is Interface. which value defines the interface that will provide the current value for such variable. The value for such variable. the same as for any other variable in this system, can be set to be evaluated at set intervals, or it can be set to its value having a value that expire after a set validity interval, When the inference engine seeks the value of said variable, either at the preset update intervals or when such value is needed but has has expired, a request if set to the specified interface to provide a value. Such variables are registered in the interface. After the interface is set-up, when the start_interface function is called by the program and the connection established between the interface (107) process and the computer program (115) process, the functions executed by the interface comprise: initialize_context (initializes the connection between the program and the interface), pause_context, resume_context, shutdown_context, receive_registration (called when the program seeks to map for the first time a variable to an external data point); receive_deregistration, poll_for_data (checks periodically which registered variables need updating, retrieved data values available, packages the data into the data structure used by the return functions, and sends the data back to the program (115); get_data (called when the program requests data service for one or more registered variables); set_data (called when the program executes one or more set actions within a rule or procedure, it sends a request to the interface to set the external data point to which the registered variable is mapped, it also may call interface return functions to change the value of the registered variable after its corresponding value in the external system has been changed). The interfaces perform also many other functions related to passing objects and messages, checking data types, and several other API functions.

19. In the current implementation of this invention, methods such as procedures, rules, formulas and relations, defined in a structured natural language, may apply to a single instance or a group of instances, or to a class or a group of classes, and they may include references to connections and to connected objects. Methods may refer or apply to the values of attributes at different time points or to the behavior of variables or parameters over time. They can perform in response to: a) given events or conditions, b) at predetermined time intervals, or c) upon request from other rules or procedures. Methods can be executed in real-time, in simulated time, or as fast as possible, implementing different strategies concurrently and over extended periods of time. Arithmetic and symbolic expressions can be used independently or combined, and dynamic models may include from non-linear differential equations to logic expressions to simulate both analytic and/or heuristic behavior.

20. The following sections will also discuss various of the innovative teachings of this invention which refer to methods and tools used to retrieve and display the stored information and data, encapsulated in graphic and interactive models of complex molecular mechanisms and pathways, allowing to:

a) interactively navigate through those pathways;

b) perform queries that refer to: a) the functional modular structure of the bioEntities involved, b) the position of bioPools downstream or upstream of the bioPool taken as reference, c) to the location in cellular compartments of the processes in which those bioPools of bioEntities participate, d) any combination of the previous three; and

c) dynamically simulate the kinetic interactions between the bioReservoirs and bioProcesses, to mimic the regulation and function of cellular systems.

21. The system of this invention can be implemented layered on top of any advanced real-time object-oriented expert-system development environment or shell that support the methodologies described throughout this application, or it can also be developed in total or in part from scratch, parting from any combination of hardware and software platforms, either already existing or developed for this or other purposes, that provides the capabilities and means required for this implementation. In particular, and for the purpose of illustrating a preferred embodiment of this invention, this system is currently implemented on the Version 3.0 or Version 4.0 of G2 Real-Time Expert System (G2) shell, an object-oriented development and deployment system produced by Gensym Corp. of Cambridge, Massachusetts, and hereinafter referred to as the "Shell", which supports dynamic simulation and temporal reasoning. While many of the underlaying capabilities referred to below are common to other static or real-time expert-system development environments, some capabilities may be currently specific for G2, but equivalent capabilities can be expected to be developed by others. This Shell currently runs on several UNIX-based platforms, and it is being extended to Windows NT. For detailed information about the underlaying capabilities of this system, Gensym Corp. should be consulted. Among the the generic, domain-independent, tools and capabilities provided by the Shell, are:

a) a proprietary structured natural language, supported by a pop-up text editor, for encoding the object definitions as well as rules, procedures, functions, relations, formulas and values;

b) a graphical language, supported by an icon editor, based on a visual representation of connected icons, and the capability to associate workspaces with objects, which allows the organization of knowledge at different levels of detail in objects within objects, each represented by icons. The Shell also allows to reason about those iconic objects, their connections, and their locations in the workspace hierarchy. Distant connections can be also maintained, and reasoned about, between objects in different workspaces.

c) an inference engine, which interprets relations, functions, rules and procedures;
d) a simulator, which may compute the values for three types of linear or non-linear systems:
- dependent variables or parameters, from algebraic equations;
- discrete-state variables or parameters, from difference equations; or
- continuous-state variables or parameters, from differential equations.

e) a supervisor, which controls operations, such as task prioritization, asynchronous concurrent operations, and real-time task scheduling. Other real-time features include time stamping and validity intervals for data, history-keeping, and real-time data interfaces;
f) a set of predefined generic object classes;
g) several inference methodologies, such as data-driven, event-driven, goal-driven or time-driven capabilities;
h) a graphic user interface development environment to create:
- user controls such as action buttons, radio buttons, check boxes, and type-in boxes; and
- graphic displays such as readout-tables, graphs and charts.

i) dynamic messages, to pass information to the developer and user;
j) an inspect facility, with various capabilities used in this invention to allow developers and users to: a) show on a workspace the class hierarchies of any specified class, or all the members of any specified class, as well as the workspace hierarchy of any specified object, or the module hierarchy; b) go to any specified object in the knowledge base, displaying that object in the center of the screen; c) search for a set of objects that fit a set of specified characteristics; d) highlight or replace any word or expression within any specified class; and e) write to a file the text of the table of attributes of any named item, or of all the items upon a named workspace.
k) a describe facility allows, through the "describe" menu option of any object to find out the relations between objects or the elements of dynamic lists, making possible to go to the icon of each element of the list, which provides a very useful facility to navigate to objects selected following desired criteria used to fill the list.
l) a tracing and debugging facility facilitates such developer's tasks;
m) the capability of breaking an application into a hierarchy of dependent modules;
n) the message board manage text messages generated by the system, such as configuration error messages, prompts generated at run-time, or explanations requested by the user. The system of this invention also use those facilities to generate messages that show up on the screen to indicate a wide variety of abnormal conditions appearing during modeling or simulation, such as all those expressions used in many procedures listed in the tables as: inform the operator that "...".
o) support for distributed applications, by allowing different knowledge-base modules to be located in several workstations. Communication between distributed modules is handled transparently by the Shell's built-in network protocols. A simulation application can be distributed among several workstations when the knowledge-base is distributed as well, allowing computer resources to be adjusted to achieve the necessary performance.

22. The Shell also provides a predefined object class, called "object-definition", which iconic instances have a table of attributes that provide frames or templates to define the characteristics of new classes of objects, such as: a) the names, types and default values for new attributes for that class, which are added to, or overriding, the set of attributes inherited from the superior class; b) an icon specific for that class, which can also be inherited; and c) the stubs that define the types of connections specified for that class. Instances of the so defined classes can be created: a) programatically, in which case they are transient when created, but can be transferred to a workspace and them be made permanent; or b) by selecting the "create-instance" option from the menu of object-definition, which creates an iconic instance of that class that sticks to the mouse-pointer and can be dropped by clicking on a workspace. Additional instances can be also created by selecting the "clone" option from the menu of a previously created instance. G2 provides an inference engine that operates on information from different sources, such as: values compiled in the knowledge base, simulated values, and values received from the user, sensors or other external sources, and responds to events from the user, simulator, or external data servers.

23. Each variable has an attribute called data-server, which value determines how said variable receives its value. Options are the inference engine (the system gets the value from an specific formula, a generic formula, or a data seek, in that order), the simulator (the system gets the value from an specific simulation formula or a generic simulation formula, in that order), or interfaces to external monitoring devices, databases, or file systems. By default, in he current implementation of this invention, the variables are simulation variables, which means that their data-server is the simulator. A simulation variable may have two values: the current value may be optionally provided by inference engine or one of the interfaces, while the simulated value is provided by the simulator. Or it may have the current value provided by the simulator, in which case it has that such value. In those particular instances where the variables correspond to substances that are being monitored in the reactor, the variables are then set to receive two values, the current measured value, provided by the external

sensors through the appropriate interface, and the simulated value. The non-simulated values may be set to expire after a desired period of time, so that the inference engine seeks a new value when needed. Different part of the simulation may run at different times, as desired. The time interval for a simulation may be specified at 4 different levels, in the following order of priorities: each variable may have its own update time; each generic simulation formula may have it own time increment, which applies to the group of variables to which said generic formula applies; the simulation-model has its update interval which applies to all the variables included in said model; and finally, the whole knowledge-base has a default simulation time increment that applies when none of the other have been specified.

a) The current implementation of this invention allows the inclusion of fuzzy-logic statements, such as if (y < 6 (+-3)) /= (z > 4 (+-2)). Fuzzy truth expressions are available only in the inference engine, rules, and procedures, while the simulator only uses discrete values. Fuzzy operators comprise: is more true than, is less true than, is not more true than, is not less true than, =, and /=. The fuzzy truth values are assigned to expressions with a fuzzy truth band in parenthesis, such as: the concentration of receptor1 > 0.8 (+-0.2). The truth-threshold is set for the inference-engine as a whole. For example, if said threshold is set to 0.7, for the antecedent of a statement to be true it must be greater than or equal to 0.7.

b) The current implementation of this invention allows the sequential addition of inputs to the reactor, at a time and in amounts that are appropriate for the current status of the biochemical processes within the reactor. In the particular example below, it is expected that the concentration of receptor1 expressed on the cell surface, which is available for detection by the receptor1-sensor, is induced at some point by some other internal events that happen inside of the cell. Such internal events are the result of complex interactions, in turn induced by the occurrence of some combination of previous events, including those events induced as a result of some previous addition of some other ligand, or any other chemical or combination of chemicals, controlled by one or more other pumps controlling inputs to the reactor. This fine control based on the knowledge about the mechanisms involved, avoids, for example, to add costly ligand1 to the reactor at a time when the cells are not ready to use it, with the likelihood that it will be destroyed by other enzymes, and the risk that it will not be anymore active when the cells become ready to make use of it at a later time. It also could prevent that ligand1 may exert damaging effects to the cells when provided when cells are in the wrong state, as it has been demonstrated for various compounds that induce programed cell death when signal from said compounds are transmitted to the cell in the absence of other required signals.

24. An example of control statement using fuzzy-logic: "if the average over the last 3 minutes of the concentration of receptor1 > 0.8 (+-0.2) and the average over the last 3 minutes of the concentration of ligand1 < 0.5 (+-0.1) and the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1 then conclude that the flow-of-pump-L1 = the flow-of-pump-L1 * 1.2 and call adjust-flow-of-pump(flow-of-pump-L1)", wherein: a) the concentration of receptor1 and the he concentration of ligand1 are variables coupled to the corresponding sensors for those substances in the reactor, which are monitoring devices external to this system; b) the statement "the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1" prevents additional changes in the flow for an specified period of time, to allow the system to adjust to the last changes; and c) adjust-flow-of-pump(flow-of-pump-L1) is a remote procedure call that activates the external device that controls the pump controlling the delivery of ligand1, while passing the value of the variable flow-of-pump-L1 to it.

a) In an alternative implementation, the bioPools of the components that regulate critical cellular events, or of those that are monitored in the reactor, may have an additional attribute defined as a fuzzy logic parameter which value can then be accessed by different monitoring/control statements. Fuzzy values may take values from -1.000 true (certainty of being false) to 1.000 true (certainty of being true). The value of said parameter could be provided by a rule, or by a generic or specific formula defined by the modeler, which is dependent on either the measured or simulated value of the corresponding concentration variable, as preferred. For example, said fuzzy attribute may be called "availability", and its value could be provided by a rule, such as: "if the concentration of any substrate S > the Michaelis-constant of S / 2 (+- the Michaelis-constant of S/2) then conclude that the availability of S is true".

- This means that whenever the concentration of any given substrate is greater than or equal to 85% of its Michaelis-constant, with the truth-threshold set to 0.7, then said substrate is considered to be available (availability is 0.7 true) under this setting. If the concentration of said substrate is smaller than 85% of its Michaelis-constant then it is considered to be not available (availability is less than 0.7 true), and if the other conditions have been met, it would cause the flow-of-pump-S to be increased. For example, the monitoring/control rules would refer to such fuzzy logic variables being true or false, and would allow the modeler to modify the rules for single or groups of bioReservoirs, without the need to modify the control rules, such as: "if the average over the last 3 minutes of the availability of receptor1 is true and the average over the

last 3 minutes of the availability of ligand1 is not true and the average over the last 3 minutes of the flow-of-pump-L1 = the flow-of-pump-L1 then ... "

- In addition, the current implementation of this invention allows to:
- a) use the simulated Virtual Model to entirely take control of when and how the inputs are added to the reactor, (or the outputs, such the culture medium or cells, removed);
- b) synchronize the simulated model to the processes in the reactor, by allowing certain critical measured values from the reactor to provide the values for their corresponding variables in the simulated model, which are then used by other dependent variables and propagate down the simulated model. Such synchronized simulated model is the used as described in a).

B. **The Virtual Model : Domain Specific Knowledge Representation**

1. The computer system, methods, and interfaces that are the objects of this invention can be applied to visually model and quantitatively simulate any type of complex system involving any type of processes and their participants. The set of palettes of prebuilt building blocks provided should reflect the needs of the particular domain in order to facilitate the tasks of the modeler, permitting the basic paradigm of "Clone, Connect, Configure and Initialize" and other automated methods. A library of prebuilt query panels and other panels, such as navigation, simulation and experiment panels facilitate the tasks for users to rapidly extract and dynamically use the knowledge and data contained in each specific application. The description that follows is based on one particular domain, that of biochemistry, to illustrate one of the implementations of such computer system, methods, and interfaces. However, similar computer systems, methods, and interfaces can be implemented in a similar manner for domains other that of biochemistry. For example, in a business domain, what in this description is called bioEntities can be substituted with different types of entities, such as employees or projects, and bioProcesses can be substituted with the different activities in which both interact, the bioReactants with the roles they play in such activities, which frequently may involve other participants. Employees are like enzymes, with the more capable having higher rate-constants. Projects are like the substrates, and employees may have higher affinities for some projects, with a likelihood that they will interact in a process in which a quantifiable measure of the outcome, such as quantity or level of quality, will depend on quantifiable measure of the employee, lets say amount of time put on that project and level of specific experience. The bioPool of each employee could represent his/her time, and the max-amount could be set to a day, a month, a year, or any other amount. The Contribution to each project would be the amount of time invested in that role, while the different activities will compete for the time left available.

2. In the remainder of the description of this invention, the focus will be on biochemical systems, which are complex, hierarchical, heterogeneous and non-linear systems, involving an interplay between the processes of transport, reaction and conformational change. They are regulated by cybernetic flows of information. The system object of this invention uses a kinetic approach to model these chemical and biochemical systems, rather than a thermodynamic interpretation, although thermodynamic variables are also included to allow modification of the behavior of the kinetic system. The current approach, which focus on the amounts of entities, such as concentrations, densities, scaled-amounts, or other quantities of each bioReservoir's bioPool of cells, molecules, or molecular complexes, in a particular state, as a continuous function of time, and on specific coefficients or on rate constants that, in conjunction with the current values of those amounts, define the velocity of a bioProcess's bioEngine, and the rates of consumption of certain bioReactants that participate in such bioProcess or the rates of production of its bioProducts, which are dependent on such velocity and specific stoichiometric coefficients, and which are equivalent to the rates at output and input from their connected bioPool, respectively. This kinetic approach is closer to the way of thinking of biochemists and molecular biologists.

3. The cellular representation in this invention aims to reproduce the fact that function is not inherent in any one component of the cell, but rather emerges from the cooperation of many components, which depend on compartmental organization and functional relationships, and deals with multiple levels of biological and biochemical structural complexity, such as physiological systems, organs and their compartments, cells, subcellular organelles, and molecules. The schematic representation on the left side of FIG.2 shows an abstraction of the types of molecular and cellular relationships and interactions that are modeled and simulated, with unlimited levels of **encapsulation** being an important feature exploited by this system, allowing to start dealing with a topic with a high-level representation, and then successively "clicking" into objects of interest, to show the next level of detail. This high-level example, at a low-level of detail, can be summarized by what a biomedical expert could describe as "Oncostatin M (201) is a biological response modifier which induces different responses on different cell types: a) it induces certain classes of cells (202) to secrete the cytokine IL-6 (203); b) it induces differentiation of primary, cobblestone-like AIDS-KS cells (204) into mature spindle-like AIDS-Ks cells (205), which in turn are capable of long-term growth in agar when c) both

Oncostatin M and IL-6 are present". This statement and its graphic representation combine knowledge at very different levels of detail, equivalent to the way human mind can abstract knowledge. For instance, it deals with both molecules and cells as cell-entities as if they were similar types of structures, when in reality that is very far away from the reality. Cells are composed of thousands of different types of molecules. In some cases, the overall pathway within the cell can be known to some extent while in others the mechanism of a biological response may be totally unknown.

4. The system of this invention can be described in very similar terms. On the one side, bioPools of cells and molecules can interact with each other, in any combination in a variety of bioProcesses, as shown in FIG.32 and described in more detail later. On the other side, as schematically shown on the right side of FIG.2, the components of each type of cell can be modeled and analyzed at increasing levels of detail, as contained in various layers of bioModels (202 - 212, 216-217), until at the end of the layer hierarchy, the single bioProcesses (214) are represented, and their associated bioReservoirs (215), which point to their associated bioEntities (not shown). In general, the signals transmitted by the cellular signaling or regulatory pathways progress within each cell in repeated cycles, driven by external and internal events, and sometimes time in a synchronous way, and they may extend from cell to cell, some times activating the producer cell as well. An typical but simplified high-level example of a cellular signaling pathway driven by a certain agonist, such as Oncostatin M, is shown in FIG.2 as: agonist (201) -> receptor activation (206) -> signal transduction (207) -> gene transcription (208) -> mRNA processing (209) -> protein synthesis (210) -> secretory protein processing (211) -> transport -> (receptor activation). By clicking again and again on objects at each level, one can focus on specific areas at the desired level of complexity. In the one example shown in FIG.2, this multiple encapsulation process has been abstracted. It illustrates a cell-bioProcess in which, after a cell (202) has been activated by Oncostatin M (201), the secretion of interleukin IL-6 (203) is induced, but only after the sequential activation of numerous biological processes (bioProcesses) (214) , involving numerous pools (bioPools) (215) of biological entities (bioEntities), contained in bioReservoirs, which are encapsulated in this simplistic example within lower-level bioModels, each representing one the major group of steps of the pathway mentioned earlier (206 through 211). Clicking on any of these bioModels, such as DNA-transcription (208) or secretory-protein-processing (211) displays their subworkspaces (212 and 216 respectively), which may in turn contain a set of other bioModels (such as those superior to 213 and 217), and the process can be repeated any number of times, until reaching bioModels composed of sets of bioProcesses (214), and their related bioReservoirs (215).

5. To represent biochemical systems it is necessary to refer to the characteristics and current state of single entities, and at the same time to refer to populations of entities with similar properties. In a small application for molecular genetics, Karp (1989) was only able to represent and implement the overall system either as a network of molecules interacting in a pathway or as a network of variables quantitatively representing the processes in a pathway, as separated systems, but he was not able to integrate both, the description of participating entities and the quantitative processes. This integration is one the teachings of this invention. Furthermore, contrary to the system of this invention, his representation was static and did not include temporal reasoning, having all processes represented as instantaneous reactions.

6. FIG.3 shows how the intermediate submodels are preferentially organized in two types of superior compartments of a cell represented by the different subsequent cell states represented by the cell-phases (302 through 308), and within each of those the cell-compartments (309 through 316) that represent the different cell organelles. While each of the cell-phases contain a detail description of large numbers of components, as encapsulated in the different sublayers, it has in addition a set of characteristic parameters and time-variant attributes (317 through 320) that can be used to model the progression of a population of cells through those compartments. That concept is schematically represented in FIG.4, where some of the cells of that population X1 may keep cycling through those phases, as represented by the bioPools of cells in each phase: Go (404), G1.1 (406), G1.2 (409), S (411), G2 (413), and M (416), if some specific and required conditions are met, as specified by their corresponding transitions represented here by the translocation-bioProcesses (405, 408, 410, 412, 415, and 417), or they can exit the cycle if other set of conditions are met (407) and differentiate to another related cell type characterized by different functions and therefore a different cell cycle (402) or if yet other set of conditions are met (407) they may dye by programmed cell death or apoptosis (414).

7. A cell-bioModel is represented as a set of bioReservoirs and bioProcesses, systematically organized in the subworkspaces of other bioModels contained in such cell-bioModel, which represent each a separate compartment in time and location, and within those, sets of biologically related processes or pathways. At a higher levels of organization, bioProcesses connected to bioReservoirs of cells or cell-interactions, are organized into bioModels the represent organ compartments, which in turn are organized into organs, and which in turn are organized into into physiological systems. Different cell-bioReservoirs contain either populations of different cell subsets or populations of the same cell subset in different stages or in different

locations. Each of those cell-bioReservoirs makes reference to a predefined cell-bioModel, which has a characteristic phenotype representative of the cells contained in that cell-bioReservoir. The different phenotypes may characterize: a) different cell lineages, b) different stages of differentiation within the same cell lineage, or c) cyclical changes in the cell characteristics over time, within the same differentiation stage. Phenotypes are sets of characteristics measurable on intact cells, and the transition of cells from one phenotype to another are represented in the present embodiment of this invention as a translocation of a fraction of the cells from the cell-pool of the former subset to the cell-Pool of the later subset. The events that trigger those transitions may be unknown, but they may be recognized by changes in measurable or functional markers, such as the appearance of a new receptor, the synthesis of DNA, the secretion of specific interleukins, or mitosis, and the relative time in which they appear or disappear may be known, in which case those transitions are time-driven. More specifically, a cell-bioEntity, which is a high-level external representation or description of a cell, may make reference to a cell-bioModel, which as described above is internally characterized by its components and mechanisms, represented by the pathways of bioReservoirs and bioProcesses.

8. FIG.5 is a diagrammatic introduction to important concepts upon which much of the processing and reasoning of the current invention is based, where the different layers of arrows represent the direction of flow of information and data. In this system, pathways refer to sets of alternating layers of bioReservoirs and bioProcesses connected to each other to indicate participation of the populations or pool of chemicals represented by the bioReservoirs in the processes represented by the bioProcesses, and to further indicate that the processes represented by the bioProcesses provide inputs to the pool of chemicals represented by the bioReservoirs. The result of those specific connections is a multidimensional network of interconnected biochemical pathways that determines the basic architecture of this system. That basic architecture also defines the resulting network of parameters and variables, and the specific arguments for the generic formulas that provide the values for those variables are determined by those connections. A bioReservoir is an iconic knowledge-structure that contains a bioPool (501), which refers to a population of a single type of molecule or complex contained within a limited space represented by the bioReservoir. A chemical process is represented as a bioProcess, an iconic knowledge-structure that contains various components, including a bioEngine (504) that controls the Velocity at which the input(s) of the bioProcess, the bioReactant(s) (503), are consumed and the output(s), the bioProducts (505), are produced. BioReactants can also be said to represent the participation of certain number of units of a bioPool as the input to a given bioProcess, and more specifically, the role that such number of units of the bioPool play in such bioProcess. BioProducts can also be said to represent the participation of certain units of a bioPool as output from a given bioProcess Both types of knowledge-structures integrate bioProcesses with bioReservoirs in a complex architecture that allows to model the kind of complex systems and multidimensional pathways that are characteristic of the domain of this invention. In such integrated system, units from the bioPool of a consumable bioReactant, are transferred to the bioPool(s) represented by one of its bioProduct(s), at a dynamically computed rate. The term upstream is used hereafter to indicate those bioReservoirs, bioProcesses, or pathways that affect the inputs to a given bioReservoir or bioProcess, while the term downstream is used hereafter to indicate those bioReservoirs, bioProcesses, or pathways that are dependent upon the outputs to a given bioReservoir or bioProcess.

9. The fast and short-term regulation, such as the temporary inhibition or activation of enzymatic activity is modeled through separate bioPools of the more active and less active (or inactive) forms. Each modification of molecules or complexes that results in qualitative or relevant quantitative changes in their activity or function, is represented as transfer of units between those different bioPools. Examples of those modifications include post-translational modifications of proteins, including allosteric changes, such as phosphorylation, isoprenylation, and so on. The slower, long-term, regulation of enzymatic activity is modeled by induction or depression of protein synthesis, which optimizes its concentration for the newly required function. Constitutive enzymes and receptors are considered to be synthesized and degraded at a constant rate, resulting in a constant steady-state level. In regulated molecules environmental signals, such as the extracellular availability of a hormone or growth factor, for instance, may cause the rate of synthesis or expression of new surface receptors to increase X-fold. If the rates of outputs are not concurrently and equally regulated by the same factor, then a new steady-state level will be reached, which may or may not return back to normal after the activating signal ceases. Examples of those modifications include: a) gene mutations and other modifications, b) DNA transcription, c) post-transcriptional splicing and other modifications of RNA, and d) translational regulation

10. BioPools, bioReactants, bioEngines, and bioProducts are classes of basic bioObjects that further encapsulate a variety of knowledge structures, including quantitative attributes, parameters and variables, which values, provided by simulation formulas, can be dynamically simulated in real-time. The amount of such units comprised in a bioPool, represented by a variable such as Concentration or other appropriate amount, is dynamically computed at run time, parting from an initial value or basal-amount set by the modeler, or in its

absence, from a default value set by the developer. The arguments of the variables of these basic knowledge structures are the output values of certain other variables and/or parameters that are attributes of either: a) the same bioObject instance or b) instances of connected bioObjects of different classes. For instance, an argument for the Consumption-Rate of some bioReactants (503) and for the Production-Rate attribute of any bioProduct (505) is the value of the Velocity attribute of the connected bioEngine (504), while the values of the Contribution attribute of each bioReactant may be arguments of the Velocity attribute of the bioEngine. Furthermore, the Production-Rate of some bioProducts (506) and the Consumption-Rate of some bioReactants (507) are arguments for the Input-Rate and the Output-Rate attributes, respectively, of the bioPool (501) to which they are distantly connected, while the argument of the Contribution of any bioReactant (503) may be the value of the Concentration attribute of the connected bioPool. Told in a different way, note that: a) the flow of data between the bioPool and a bioReactant is bidirectional, involving two variables in the bioPool and two in the bioReactant, b) the flow of data between a bioReactant and the bioEngine is bidirectional, involving two variables in the bioReactant and one in the bioEngine, c) the flow of data between the bioEngine and a bioProduct is unidirectional, involving one variable in the bioEngine and one in the bioProduct, and d) the flow of data between a bioProduct and the bioPool is unidirectional, involving one variable in the bioProduct and one in the bioPool. The reversibility of any binding can be represented in two alternative ways: a) implicitly reflected in the output of a single process, which is the balance in the predominant direction (as represented by the specific or global kinetic parameters), or b) explicitly modeled as two separated processes in the forward and reverse directions.

11. The value for the basal-concentration, basal-density or basal-scaled-amount parameters of each population reflects the physiological steady-state value, when the rate of change is equal to 0 or equivalently when the sum of the inputs equals the sum of the outputs. The bioPools representing synthetic agonists, antagonists or inhibitors, such as drugs or toxic substances, have a basal-concentration of 0.0, have no modeled inputs other than the user-entries, and therefore the Σ inputs = entry value, at the compartment where it is first introduced. However, it can have several modeled outputs, including the participation in modification-processes and translocation-processes, in addition to the decay term that represents diffusion, degradation and dilution. It is further constrained in that the $[As]^*(\mu+\Sigma \text{ outputs}) \leq [\text{entry}]$. Concentrations are used in general within classes of bioPools representing soluble entities. However, as it is found empirically in biological systems, most most entities are associated with membranes, with large polymerized structures such as the cytoskeleton or the extracellular-matrix, or associated with one or more other macromolecules forming complexes. This means that the concentration of those bound entities is less relevant than the actual amount of those molecules that are available in the appropriate compartment at a given time, and therefore the term density is used to represent units per compartment. Furthermore, most simulations of biological systems are more meaningful in terms of relative amounts or ratios among the quantities of interacting or regulatory molecules, and therefore the alternative variable called scaled-amount is provided for each bioPool, which is dimensionless and therefore do not require units. The values of the Velocity of the generic bioEngines, as provided in the palettes, are computed using the default generic simulation formulas associated with the subclass of parameter or variable associated with it, which by default are dependent on the values of a set of scaled variables and parameters. The user can however replace them with instances of other velocity-pvar subclasses from the library provided (FIG.11), each with its associated generic simulation formula based on absolute values (Table 85), depending on the subclass.

12. Live and growth depends upon closed cycles of mutually dependent interactions. In a constant environment, the proportions of the various constituents settle down to constant values and a steady-state is reached. The steady-state correspond to an optimum state, since the lack of such balanced state would lead to rate-limiting steps. When the environment changes, those proportions move towards new values to achieve again optimum growth in the new environment. The principle underlaying this system's dynamic modeling is the network of a combination of state and dependent variables, encapsulated within the structure of the bioObjects contained in the specified bioModel. The bioObjects provided are programmed by default for steady-state modeling. A dynamic simulation is initiated after the introduction of desired perturbations or initial conditions by the user, and the input data initiates a forward chaining which involves both control and data flow from bioReservoirs to its connected downstream bioProcesses, and from these to their connected downstream bioReservoirs, moving along the bioModel's pathways. Inputs from external control systems or databases can be also forward-chained during run-time. A variety of methods are then used to compute or infer new values for the variables or parameters, derive conclusions and pass on control signals, and trigger action sequences, each as appropriate. The forward and backward associations between bioObjects for runtime execution are either inherent in the connections between the bioObjects or are explicitly configured through the model-blocks. The required integration of dataflow and sequential control mechanisms is accomplished in the currently preferred embodiment while taking advantage of the intuitive capabilities provided by the graphical architecture, where the bioObjects encapsulate the data to which related methods apply, and the parameters and variables are

hidden but their values can be displayed. The specific way the bioObjects are connected specify both data flow and control flow, representing sequential or concurrent ordering of procedure execution, and the information needed to execute an algorithm is provided by or inferred from the architecture. The evaluation methods of the bioObjects involved in the simulated pathways execute in parallel during forward chaining, passing along the arguments to the posterior bioObjects, while the arguments of the anterior bioObjects are passed to them.

13. There are other parameters and variables defined in the system of this invention, which are also involved in the dynamic quantitative simulations of the visually modeled systems, and which can be incorporated in two distinct and alternative forms of simulation reasoning, absolute or scaled, as will be described in more detail under the heading Simulation Mode. For example, the factors that affect and define the transformation rates, such as the catalytic, binding or transfer rate of a bioProcesses (here preferably encapsulated as the Velocity attribute of its bioEngine), are distributed as attributes of the various connected bioObjects and comprise:

a) the quantities of the bioReactants of different types, such as enzymes, substrates, inhibitors, and activators, or receptors, agonists and antagonists, or components of a complex (here preferably encapsulated as the Concentration or Density or Scaled-Amount attribute of the bioPool connected to each bioReactant, which values can be directly incorporated into the equation that gives the value of such Velocity, when using the absolute reasoning; or indirectly incorporated into the equation that gives the value of the Contribution attribute of each bioReactant, which holds an intermediary scaled value that is incorporated into the equation that gives the value of such Velocity, when using the scaled reasoning);

b) the values of specific kinetic parameters characteristic of each type of bioReactant (here preferably encapsulated as: the "effective-binding-sites" attribute of enzymes and receptors, and the "Michaelis-constant", "equilibrium-dissociation constant", or equivalent attribute of substrates, ligands, or other bioReactants, respectively, which values can be directly incorporated into the equation that gives the value of such Velocity, when using the absolute reasoning, or, alternatively, the "scaled-Michaelis.k", "scaled-equil.dissoc.k", or equivalent attribute of substrates, ligands, or other bioReactants, respectively, when using the scaled reasoning), or, alternatively, a global kinetic parameter specific for that bioProcess that replaces and represents the combination of the specific parameters of each of the bioReactants of that bioProcess (here preferably encapsulated as the "rate-constant-sec" attribute of its bioEngine);

c) the stoichiometric coefficients (here preferably encapsulated as the "stoichiometric-coeff" attribute of bioProducts and bioReactants); and

d) the effects of environmental conditions such as temperature and pH are also specifically defined as attributes, and can be integrated into the simulation within the appropriate equations.

14. Chemical and biochemical reactions are continuous processes, within the time intervals that are dependent on the type of reaction and the environmental conditions, with the balance of the reaction favoring in general the forward direction. The strategy used in the system of this invention is to forward chain from inputs values from the user-interface or from databases. In the preferred embodiment of this invention, the specific way the bioObjects are connected specify the sequential or concurrent ordering of data flow, and the parameters of model equations are included as object attributes. Model-generated results may be used as input data for other simulated equations or as events to be used by the inference engine. These dynamic models can also reason about the behavior of variables in the past and project the behavior of variables into the future. Integrating the activities of the different parts of a model system requires the simultaneous solution of a set of non-linear ordinary differential equations and a set of dependent algebraic equations. In this invention, this system takes advantage of the two methods of explicit integration supported by the Shell: either the first-order Euler scheme or the quadratic fourth-order Runge-Kutta scheme, depending of the desired accuracy of the computation. The updates of the different variables may be synchronous or asynchronous, as a result of the variables having equal or different update intervals. The dynamic values of variables or parameters may be displayed in both digital and graphic forms.

15. The default values and equations provided with the preferred embodiment of this invention, represent an steady-state modeling approach, since biochemical systems are better represented by steady-state than by equilibrium situations. The value for the Basal-Concentration of each population reflects the physiological normal steady-state value, when the rate of change is equal to 0 or equivalently when the sum of the inputs equals the sum of the outputs. Enzymes that are constitutively abundant and in their active form have no effect on the steady-state, since they are not rate limiting. However, regulated enzymes, such as kinases and phosphatases among others, may be constitutive only in their inactive form, and their Concentration influence the Production-Rate of the active form in the same way as the Concentration of a substrate controls its consumption rate. In the system of this invention, the activation-process of an enzyme (u other bioEntity) means consuming units of the bioReservoir of the inactive form and producing units of the bioReservoir of the active form, and therefore it is the same treatment as for any other substrate and product. The regulation of enzymatic activity is modeled through separate bioProcesses that represent forward or feedback inhibition or activation

processes or pathways. Each modification of molecules or complexes that results in relevant qualitative or quantitative changes in their function, is represented as separate bioEntities, which in turn are represented in separate bioPools and bioProcesses. Examples of those modifications comprise: a) gene mutations and other DNA modifications; b) post-transcriptional splicing and other modifications or degradation of RNA; c) post-translational modifications of proteins, including allosteric changes, phosphorylation, isoprenylation, and so on, and degradation ; and d) modifications and degradation of other molecules. Synthesis and degradation, as well as allosteric changes or other modifications, may also be implicitly modeled within the parameters that affect the Velocity of a bioProcess, or as separate mathematical model-blocks connected to bioPools as inputs or outputs.

16. The types of inhibitions in which a reversible or irreversible complex is formed, such as the binding of an antagonist to a cell-receptor, and the feedback activation or inhibition by secreted products, are all modeled as separate receptor-processes or complex-formation-processes which compete in the background for the amount of cell-receptor available in the common bioPool, according to their respective affinities and concentrations. The reversibility of any binding is usually not explicitly modeled, unless the complex is a target for regulation, and it is implicitly included in the output which is the balance in the predominant direction (as represented by Ks vs. k_1 and k_{-1}).

17. In fast growing cells or organisms, the dilution factor resulting from cell growth is represented by a parameter μ, while in slow growing cells direct enzyme degradation is more important, since the levels of many enzymes in animal cells is controlled by the balance between enzyme synthesis and degradation, called turnover. The synthesis of an enzyme is represented differently, depending on the available information or the level of detail desired, as a zero-order process or as a first-order or higher-order kinetics, proportional to the concentration of other enzymes or regulators that participate in the complex processes of, while the degradation may have a first-order kinetics proportional to the concentration of enzymes, or or higher-order kinetics, depending also on the concentration of other enzymes or regulators that participate in the regulated degradation process. The total response associated with any particular population of receptors is represented as the maximal activity of a single receptor in the activated state times the receptor density, or scaled-amount, of the bioPool representing that particular compartment.

18. The system of this invention deals with the "incomplete information" characteristic of biological systems by integrating the scaled or semiquantitative values with the absolute values, as discussed in later sections. The scaled-valued variables, such as scaled-amount, have values within the 0.0 to 100.0 scale to normalize the diverse ranges of magnitudes involved in the system. The default initial values of the scaled-basal-amounts vary within this range to best represent the knowledge about the physiological steady-state conditions. As a way of examples: a) a value of 100.0 may model an constitutive abundant enzyme, receptor, or complex-subunit in their inactive form, while this value may decrease at run-time as the value(s) of their active or other derivative(s) increase, so that the total remains 100.0; b) a value of 50.0 may model the normal steady-state catalytic concentration for a constitutive regulated active state of an enzyme, or the steady-state concentration of a substrate, binding protein or ligand, or c) a value close to 0.0 may model any highly regulated induced molecule.

19. **Temporal reasoning** is important to model and describe the causal mechanisms that drive biological systems. Temporal reasoning is achieved in the system of this invention in several ways:

a) In general, temporal reasoning is implicit in this real-time system during a simulation, when the values of the variables and many of the parameters vary over time.

b) Entire contents of BioModels, such as those representing the successive phases of the cell cycle, and other time-submodels within those phases, can be activated and deactivated over time by activating or deactivating the subworkspace of such bioModels, as specified within procedures and rules, at pre-specified simulation time intervals, or based on events generated at run-time, such as a certain variable reaching a threshold value, or a combination of both.

c) The subworkspaces of specific bioProcesses can be also programmed to be activated for a specified period of time after their activation, as given by its activation-hold-interval attribute. This attribute can be a default value or a value entered by the user or as modified at runtime. This is an alternative to having the control of the inactivation of such subworkspace based on another event. It can also be reasoning by a combination of a given time interval or a given event, whatever comes first, or by events triggering the change of the value of the activation-hold-interval attribute in a pre-specified manner. One such option is to provide a tabular function such as by looking.

20. In the currently preferred embodiment of this invention, a unit of a given chemical entity, referred to as a bioEntity, which describe the characteristics and state of a single molecule or complex, is considered a separated knowledge structure which is referred to as an attribute of a bioReservoir and can be accessed from

the menu choices of this and other bioObjects, such as a bioReactant connected to its bioPool, which refers to a population of molecules with the characteristics described for such bioEntity. The information about the characteristics of particular functional states of a given molecule or complex is encoded as separate instances of classes of bioEntity, as will be described in detail in later sections. Different behaviors of each bioEntity are then explicitly defined through the mechanisms represented by the bioProcesses in which bioReactants that make reference to such bioEntity participate. In other words, it can also be said that bioProcesses represent the interactions of a bioPool of such bioEntity with bioPools of other bioEntities, as defined visually through their connected bioReactants or bioProducts. Furthermore, in the present invention, a single molecule can be described in more detail by and ordered and connected set of instances of more elemental bioEntities, referred to as bioEntity-components, which represent molecular functional components such as subunits, domains, motifs, active sites, regulatory sites and mutation sites. The additional levels of detail are implemented in this invention through the **encapsulation** of subworkspaces. In a similar way, sets of simple molecules can be components of other bioEntities representing a complex, or of a heter-mol-complex. Since unlimited levels of encapsulation are allowed, it is possible to start dealing with a topic with a very general visual overview, and then successively "clicking" into objects of interest, to show the next level of detail.

21. As will be described in the following sections, in the system of this invention, the methods used to process the application-specific knowledge are separated from the application-specific knowledge itself. The domain-specific but application-independent knowledge is encoded in at least four different forms, characterized by: a) the selection, combination and specific connections of the simple knowledge structures that contain the variables and parameters that quantitatively define this system, within the subworkspace of a prebuilt composite bioObject; b) the selection and design of those simple structures and their interactions as components of the composite structures, which provide for the resulting transparently interconnected architecture underlaying this system; c) the selection and definition of the specific attributes of each of those types of structures, which describe and characterize the potential behavior of those structures; and d) the methods in the form of procedures, rules, formulas, functions and relations that together provide for the monitoring, control and execution of the system. On the other hand, the application-specific knowledge is incorporated in at least two different ways: a) the selection, combination in different ways and connections of those prebuilt composite bioObjects and predefined structures; and b) the configuration of the values of the predefined attributes of those selected structures.

C. Operating Facilities for Developing, Modeling, Navigating, Queering, and Simulating the Virtual Models

The following descriptions of objects, tasks, procedures, and the complex processing that follows only highlights some features, and is not meant to substitute for the more detailed descriptions shown by the pseudo-code listings provided as Tables in the Appendix, as referred to within the text when appropriate.

1. Developer Mode: Definition of the Building Blocks

a) Domain-menus

- The **Developer Mode** can be restricted to only those users with development rights, in which case it is the default mode when the application is started. FIG.9 shows the domain-menus associated with this mode in this invention, which are shown here in an expanded way to facilitate discussion of the organization of the definitions of object classes and methods. The options shown highlighted with white background are selected options. Selecting the "Help & Mode Menus" head (901) pulls down a set options, each causing upon selection specific actions, including displaying named workspaces or subworkspaces of named objects, such as panels, palettes or libraries, or generating other walking menus. In addition, selecting any of the available user modes changes the mode of operation and displays the specific domain-menus associated with that user mode, both actions applying only for the window from which it is selected. Selecting Developer Mode (902) displays the additional menu heads: "Object Definitions" (903) with pull down options pointing to major groups of object classes, "Variable Structures" (909), "Rules Proc's & Relat's" (913), and "Formulas & Functions" (916). Domain-menus include walking menus, and whenever an option has additional suboptions then a new pull-down menu is created and displayed in the next adjacent menu slot. For example, selecting "bioEntities" (904) displays the pull-down menu headed by "bioEntities" (905), which options refer to different subclasses of the class bioEntity, and further selecting "MoleculComponents" (906) displays the one headed by "MoleculComponents" (907), and further selecting "Protein Motifs" (908) displays a workspace (not shown) that contains the definitions of the class protein-motif and all its subclasses. In a similar way, selecting "Variable Structures" (909) and further "Variables" (910) displays the menu headed "Variables" (911), and further selecting "Velocity pVariables" (912) displays the workspace shown in FIG.11 containing the definitions of the class velocity-pVar and its various layers of subclasses. In other cases options bring directly to a workspace, such as for "Rules Proc's & Relat's" -> "GeneralQuery Proc's" (914) which directly displays

the workspace (915) that contains the definitions of some of the procedures needed for the processing of General Queries.

b) **Descriptions of General Groups and High-Level Classes**

- The classification and definitions of the object classes is now described. Table 1 shows the different classes of objects and groups of tools and methods comprised in this invention, which are all knowledge structures implemented as objects. Those tools and methods are preferably organized into major groups, each comprising several object classes, which may each further comprise several levels of more specific subclasses, to be discussed in more detail in the following sections. The class bioTool (Table 6), a subclass of the class object defined within the Shell, is defined to include a variety of auxiliary iconic structures used in this invention, and has no additional attributes. Each top class of those groups is a subclass of one of the generic classes of objects defined within the Shell, from which they inherit a set of attributes and capabilities. Several of those capabilities may be proprietary technology specific for that Shell. For further details, the reader is referred to the C2 Reference Manual Version 3.0 or Version 4.0 and other documentation produced by Gensym Corp., which are here included by reference. In this filing, only those Shell-defined attributes that are necessary to describe this invention will be considered, and these attributes or their equivalents may be common to other development environments. Other attributes and capabilities not mentioned here may, however, facilitate the development tasks and improve the performance of this invention. In the following discussion, attention will be focused mainly on those attributes incrementally defined for each subclass.

- There are several other basic classes defined within the Shell that the system of this invention uses without relevant modifications. One of those are **workspaces**, which are items used to hold and display other items, and which attributes include names, user restrictions, margin, border, background, and module assignment. Workspaces can be independent higher-level workspaces, or subworkspaces subordinated to objects. Subworkspaces may also have the capability of being activatable, if so defined in their superior object, which means that the objects upon them are only active and accessible to the inference engine and the simulator when such subworkspace is activated. Other capabilities of the Shell, such as flashing of the icons, changing colors, or rotation of specific patterns can be used to animate a simulation and to indicate relevant events, such as showing the bioObjects that are activated, or which variables' values are out of range, and many other applications.

- The object class definitions listed in Tables in the Appendix include only those attributes which provide new and relevant information, such as: a) "class name", b) "superior class", and c) "attributes specific to class". Other attributes are listed only when relevant or modified, following the following rules:
  - "**Capabilities and restrictions**" describes which additional capabilities or restrictions apply to all the instances of all the subclasses of that class. The Shell-defined capabilities used in the current implementation of this invention comprise: "**activatable-subworkspace**", for the classes bioReservoir, bioProcess, time-compartment, and hold-bin; and "**subworkspace-connection-post**" for the class model-box.
  - "**Class restrictions**" defines which menu options and other non-menu choices are available or disallowed, or which attributes are visible in the table, in each particular user mode for all the instances of all the subclasses of the class for which they are defined. The Shell-defined non-menu choices include: "move-object", ... The Shell-defined menu choices include: "table", "name", "transfer", "create-subworkspace", "change-size", "describe", "disable", "delete", "rotate-reflect", ...
  - "**Inherited attributes**" are usually not be listed on the Appendix, since they can be deduced by adding up the listings of "attributes specific to class" of each of the superior classes. On occasions, however, inherited attributes may be listed for the purpose of summarizing disperse definitions. When an attribute is defined in more than one object-definition, the most proximal (or lower-level) definition overrides the others.
  - "**Menu option**" refer to whether the name of the object class appears as an option within the new-object selection from the menu of workspaces, or with other words, whether the class can have instances (if the value of this attribute is "a final menu choice") or whether the class is only a superior to other subclasses (if the value of this attribute is "not a final menu choice"). This attribute will be only listed for those high-level classes which are "not a final menu choice", and it is otherwise assumed that the class is "a final menu choice".
  - "**Stubs**" define each of the connection stubs for that class and their position on the class' icon, and optionally the direction of flow and/or the name of the connection port. The stubs can be inherited from superior classes, in which case the value of this attribute will be inherited, or the class may have no defined connections and the value is none, and in either case this attribute may not be listed. If it is not listed and the superior class have defined stubs, then it is assumed that the stubs are inherited, and if the superior class have no defined stubs, then it is assumed that the value of this attribute is none.
  - "**Icon description**" defines the iconic appearance of the icon characteristic of all the instances of the class. The icon can be inherited or be specific for that class, and usually contain differently colored layers, or

:olor-regions, which color may change dynamically at run-time, to visualize the state of the objects, such as selected. or of their subworkspaces, such as activated. The detailed description of the icons is not included in the Appendix to save space, and in most occasions it will be simply referred to as the <name of the class>-pattern, followed by its dimensions (to have a reference for the stubs), and by the color-regions, which may be referred to by rules and/or procedures.

- In addition to an attribute table and sometimes a subworkspace, each iconic object has an associated menu with a set of options. Some of those menu options, and options defined within the Shell, perform standard tasks common to all objects , such as: change-size, clone, color, delete, describe, disable, go-to-subworkspace, move, and name. Many of those tasks are self-explanatory and will not be further described. Other novel and domain-specific menu options are defined in this invention as **user-menu-choices** to provide interactive access by the user to important automatic or semi-automatic tasks related to the configuration of the knowledge structures and to requests diverse forms of processing and display of such knowledge. User-menu-choices are defined for the specific class named in its applicable class attribute, as in the example shown in Table 27. They appear as options in the menu associated with the icon of every instance of every subclass of said applicable class, unless a class restriction is defined for any of those classes to exclude that option from the menu in any particular user mode . Selecting any particular options triggers one or more actions. All those domain-specific tasks provide a highly interactive user-interface and are important teachings of these invention. Those options and the methods invoked upon their selection will be discussed in later sections. Here we will focus on the definitions of all object classes and on the description of their attributes.

c) **Variable Structures:**

- **Variables and Parameters:**

- In the present invention, the quantitative attributes of object classes may be defined as simple attributes, or as instances of subclasses of parameters or variables. The values of simple attributes can be modified through rules, procedures, or other actions of the inference engine. The values for the variable and parameters may be provided from different sources. The Shell's simulator is used in the current implementation to compute the simulated values of variables and parameter that are integral parts of the bioReservoirs and bioProcesses, as described later, including state variables. Those values are preferentially provided by generic simulation formulas, although modelers may also provide specific simulation formulas for specific variables. Users can also input values for the simulator through the input-panels described below, and the program can also update values based on values received from other sources, so that values from different sources are integrated. To standardize the units used during a simulation, the absolute values are preferentially entered in Molar, M/sec, moles/liter, or seconds, as pertinent.

- There are large number of subclasses of parameters and variables involved in this system, as defined in Tables 2 through 15. Only a few of them are used as independent parameters or variables represented by their icon, and usually they have auxiliary functions, such as the x-pos and y-pos (Table 3) used to aid in the graphic drawing of pathways on a workspace. In this invention, most parameters and variables are implemented as attributes of the various classes of bioObjects, or other objects such as model-blocks or inference blocks, as well as attributes of other variables. Aside from the high-level classes, most of the other subclasses do not have additional specific attributes, but many have distinctive default settings, as specified in their listed definitions. The separated subclasses allow to attach specific methods to each class, either in the form of formulas, or by referring to all members of the class in rules, procedures or other actions. Since it would be tedious and repetitious to discuss each subclass of parameter and variable that are referred to in the system of this invention, listed definitions of other objects, their definitions are instead listed in lumped groups by similarities, and only a few that are representative or important to the understanding of the operation of the overall system will be discussed in the relevant section.

- In the currently preferred embodiment, domain-specific variables and parameters are subclasses of various subclasses of the the Shell's class "parameter", such as float-parameter, integer-parameter or symbolic-parameter, text-parameter or truth-parameter. Those that represent parameters in this domain are grouped under the domain subclasses int-par (Table 3) or float-par (Tables 4 through 8), depending on their value types. Those that represent variables in this domain are grouped under the domain subclass float-pvar (Tables 9 through 12), which are subclasses of float-parameter. The reason for these discrepancies is that In the Shell's classes "parameter" and "variable" do not fully correspond to the common concepts of variables and parameters and are simply two different data structures with different constraints and capabilities. This invention also defines a set of subclasses of float-var (Tables 13 through 15), a subclass of the Shell's class float-variable, as an alternative for domain attributes such as Concentration, Density and Velocity (which by default are represented by pvars, as discussed in more detail in the next paragraph) or parameters such as a variety of kinetic constants (which by default are represented by simple float attributes).

- The Shell's "variables" have capabilities important for the implementation of this invention, such as: a) allowing the user to set a desired instance, lets say the concentration or density attribute of a bioPool representing an entity which quantity is monitored in a reactor, to receive a measured value from the external monitoring sensor while its simulated value is being simulated in parallel, based on the "Virtual Model" specifications, and then by using inference rules to monitor either value or both values, and to compare them to a specified threshold or range or to each other, and to initiate some control actions to regulate the reactor system monitored by the sensor; and b) allowing the user to define simulation or general formulas specific for a particular instance of a "variable" by writing the formula directly in the corresponding slot for the formula in the instance's simulation subtable, while the Shell's . Because the Shell's "variables" are larger structures that require more memory space, this invention uses by default subclasses of the Shell's "parameter" as attributes of bioObjects. However, alternative but equivalent subclasses of variables are also provided, allowing the user to delete the default "parameter" and replace it with the equivalent "variable", allowing the user to then write any customized specific simulation (values provided by the simulator) or general (values provided by the inference engine) formula to model that particular "variable". Differences between the Shell's "parameters" and "variables" include: a) "parameters" do not have slots for specific formulas but can receive values from generic formulas that apply to instances of a class; b) a "variable" may have an initial value and two current values, the inferred and the simulated values, while a "parameter" have only one value that can be either inferred or simulated; c) the inferred value of a "variable" can be set to expire after a time interval and therefore a "variable" may have no value (it should not expire when the value is used by the simulator), while a "parameter" always has a value; c) "variables" can backward-chain to seek its values, while "parameters" cannot; and d) "variables" cannot be directly referenced by procedures. When using a different Shell with parameters and variables with different attributes and capabilities, some modifications may be necessary. Examples of default instances are shown in FIGs. 13 through 17 (and their definitions in the indicated Tables) for "variables" such as: velocity-var (1301, Table 13), model-block-output-var (1428, Table 14), model-block-var (1430, Table 14), and for "parameters" such as: basal-density-par (1506, Table 7), scaling-density-par (1508, Table 6), density-entry-pvar (1510, Table 9), scaled-entry-pvar (1512, Table 9), input-rate-pvar (1514, Table 11), density-pvar (1518, Table 10), basal-conc-par (1522, Table 7), conc-pvar (1524, Table 10), contribution-pvar (1615, Table 10), consum-rate-pvar (1617, Table 11), binding-rate-pvar (1706, Table 11), time-delay-par (1711, Table 8), ph-par (1713, Table 5), ph-factor-par (1715, Table 5), and produc-rate-pvar (1723, Table 11).
- The details of a "variable" are here discussed in reference to FIG.10, where the attribute table (1001) of velocity-var shows the attribute slots provided to enter specific information to define and control each particular instance, such as the methods of inference allowed (1002); its name (1003); data type allowed (1004); optional initial value (1005); last recorded value (1006); the number of data points or the time period for keeping the values of the most recent history of that variable, together with the time stamps at which those values were collected (1007); the period of time for which each value is valid (1008); the formula slot where the user may optionally write a specific formula for the inferred value of that particular instance(1009); details about its optional simulated value (1010), which may include an additional simulation subtable (1011), which contain slots that allow the user to optionally control the simulation of that particular instance, such as the time increment for update (1012), a specific simulation formula (1013), and the number of data points or the time period of values to be stored (1014); the main table also provides for the initial value for simulation (1015); whether its value is provided by the inference engine, any simulator or external interfaced sources, such as sensors or databases(1016), and the default update interval (1017).
- To add a variable or parameter of the desired class as an attribute of a bioObject, the modeler can select the "add subtable" from the menu that appears when clicking on the slot of the value that attribute, and then navigate through successive menus to follow the hierarchy of subclasses of variable or parameter, in this case of float-variable, until the desired subclass can be selected, which results in the creation of an instance of that subclass that gives the value to the attribute from which slot it was requested. To replace an existing variable or parameter for another, the existing one is first deleted, and then the sequence above is followed. To define the specific simulation formula the same process has to be followed Some examples of such attributes comprise.
- Another example of why and how specific instances of the default "parameters" are to be substituted when needed for the larger "variables" is discussed here. The Velocity attribute (1705) of an instance (1701) of a subclass of bioEngine is an instance (1701) of a subclass of rate-pvar, binding-rate-pvar, which value is simulated according to a generic-simulation-formula that refers to such subclass (example in Table 83) and represents a mathematical model for all reactions of such type. The Velocity attribute is allowed more flexibility in this invention because there are not only many types of processes, depending on the participants in each case, but there are also many opinions about what is the most fitting equation to compute the Velocity of

each type. In addition, there are two types of simulation provided by the system of this invention, as discussed under the Simulation Mode heading. One is semi-quantitative, using scaled (dimensionless) variables, and the other uses variables with absolute values. The currently preferred embodiment of this invention is defined by default with a matching set of scaled variables, each with its associated simulation formula. For the Velocity attribute, depending on the bioEngine subclass, different subclasses of rate-pvar (Table 11) are used, such as: catalytic-rate-pvar, binding-rate-pvar, dissociation-rate-pvar, modification-rate-pvar, or transfer-rate-pvar, with such scaled-values based formulas. However, the modeler can substitute those with an appropriate instance of an alternative set of more specific subclasses of velocity-pvar (Table 12), such as those shown in FIG.11 with more specific absolute-value based formulas. Each subclass of velocity-pvar (1101), such as: enzyme-velocity (1102), receptor-velocity (1103), complex-formation-velocity (1104), complex-dissoc-velocity (1105), ion-transport-velocity (1106), conform-change-velocity (1107), and translocation-velocity (1108), has more specific subclasses, and each of those subclasses has its associated more specific absolute-value based simulation formula. The nomenclature used for naming each subclass indicates all the participating components and the reaction type. For example, 1E.2S.1I.ORD.velocity is meant for a bioengine.1E.2S.1I, which means that its icon has at the top four specific stubs connected to one enzyme.r, two substrate.r and one inhibitor.r, and the formula for that instance reflects the fact that the reaction happens in an ordered fashion, that is substrate1.r binds before substrate2.r, and the inhibitor is a competitive inhibitor of the first substrate. The subclasses of velocity-pvar, and their associated simulation formulas, represent only the more typical cases, but they are only a fraction of all possible cases that user may need to represent. Therefore, another alternative is provided for the modeler to write the simulation formula that most appropriately fits the system to be modeled, substituting with an instance of the class velocity-var described above.

- The system of this invention utilizes the history keeping capability for the values of relevant parameters and variables, as specified in their listed definitions, to graph the time-line or to reason about those historic values. The user can decide to display those values in charts or other display forms. The historic values, as well as their maximum, minimum, average values, or rates of change over time can be not only be displayed, but also used by the user in different ways, such as storing them in external files or databases, allowing the values to be further analyzed.

- **Arrays and Lists**

- Arrays and lists are structures that hold a number of item of values in an ordered way. The subclasses of arrays (Tables 16 and 17) and lists (Tables 18 through 20) have all the characteristics of their parent classes, as defined in the Shell, and they may have additional attributes within the system of this invention, as specified in their definitions. Arrays are indexed, and therefore each of its elements can be directly referred to by referring to its index, while lists may be indexed or not. The length of arrays have to be predetermined, before additional values are added, while the length of lists increase automatically, as more elements are added. However, the current implementation of this invention uses arrays to permanently store the transient values of its elements by setting the "initial-values" attribute of an array equal to the desired values to be stored, separated by commas, which may be the current values of the elements of that particular array at any point in time, or the elements of an auxiliary list, as described in the section describing the methods for queries.

- One of the classes of arrays used in the current invention is query-array (Table 31), a subclass of symbol-array, which instances are created by the query initialization procedures. For example, one of the uses of a set of query arrays is for storage in each of all the instances of each class of molecular components in the Virtual Model (Table 16), as described later. The instances of references-array (Table 32), a subclass of text-array, refer to sets of instances of the class reference-block (Table 34) and may be attributes of instances of bioEntities, bioReservoirs or bioProcesses, or bioModels, being used as described below. Furthermore, instances of the Shell's class float-array can be used to store all the values of a desired parameter or variable during a monitoring session or a simulation run, which can then be stored to a file or passed to another external program or device for further analysis or processing.

- Bioitem-list (Table 20) is a subclass of the Shell's class item-list with no additional attributes. The different subclasses of bioitem-list (Table 20) are restricted to instances of particular classes of objects, such as experiment-selections, bioReservoirs, and bioProcesses. This invention uses lists in various ways, including: a) as transient auxiliary structures to process information at run-time, not visible to the users, such as instances of query-list (Table 18), a subclass of symbol-list, are used as auxiliary structures in the creation of query-arrays, and instances of scroll-text-list (Table 19), a subclass of text-list, are used as auxiliary structures in the creation of scroll-areas; and b) as auxiliary structures with their icons upon a workspace, which in addition to be used to process information at run-time they are available to the user for inspection, such as those shown in FIGs. 31 through 35) listing upstream and downstream bioReservoirs and bioProcesses, in reference to the

bioObjects from which they where requested. Clicking on those icons by the user causes the display of the elements of the list, providing direct access to the objects listed and to the attribute table of the objects listed.

d) **BioTools**

- **Class model-block**

- The **class model-block** (Table 22), a subclass of the class bioTool, may have as many subclasses as needed to represent the different mathematical models needed by the domain modeler, and some representative subclasses are shown (Table 22) as examples. Some examples of generic simulation formulas used to compute the value of the outputs for those subclasses are shown in Table 87. Also included is a generic-model-block, to be further defined by the modeler, which is a place holder for new parameters and/or variables to be added, and for specific general or simulated formulas for those newly added variables. The additional attributes for the class model-block includes "output.1" and each subclass have additional specific attributes which are simple attributes, parameters or variables, and which vary in number and type (Table 22). A model-block is an iconic object, preferentially used upon the subworkspace of a model-box, which encapsulates the components of a mathematical model, as defined by its attributes, where the value of one or more of those attributes, the outputs, are dependent on the values of inputs, which may be constant or may changed at run time. The inputs may be have any type of origins, including: a) one or more other attributes of the model-block; b) one or more attributes of other objects in the Virtual Model; c) one or more values originating from any external source interfaced with the system, such as external sensors, databases, or external simulators; d) any auxiliary independent constant, parameter, or variable; e) any timer or meter; or f) any combination of the above.

- As shown in FIG.12, clicking on an instance of model-block (1222) displays its menu (1223). The "disable" option (1224) is used when a set of alternative model-blocks is configured for a given bioPool to disable those not currently in use. The "table" option (1225) displays the table of attributes (1226) and provides access to its variables. Two classes of variables defined to be used as attributed of model-blocks: model-block-var and model-block-output-var (Table 12) differ in the default settings: only the former has by default a simulation subtable and only the latter keeps history of its value, while both have an initial value and indefinite validity interval, so that they always have a value, and therefore can be used by the simulator. Clicking on the slot (1227) of the attribute "output.1" displays its subtable, the attribute table of an instance of model-block-output-var (1228). Although the predefined model blocks classes have associated generic formulas to compute the value of their output.1, the modeler may want to override it by requesting, by clicking on the slot of its simulation details attribute, the addition of a simulation subtable. The subtable of the attribute "input.1" (1229) shows the table of an instance of model-block-var (1230), and clicking on the slot of the attribute "simulation-details" (1231) displays its simulation subtable (1232) with additional attributes, among them one (1233) to hold the simulation formula to be written by the modeler. The particular subclass selected in this example, a proportional.f, is defined with two additional attributes given by simple float attributes with default values of 1.0 for "gain" (1234) and 0.0 for "bias" (1235), which represent what their names indicate.

- **Class inference-block**

- The **class inference-block** (Table 23), a subclass of the class bioTool, may have as many subclasses as needed to represent the different inference models needed by the domain modeler, and some representative subclasses are shown (Table 23) as examples. Some examples of generic simulation formulas used to compute the value of the outputs for those subclasses are shown in Table 88. The additional attributes for the class inference-block includes **Outcome**, given by an outcome-par, and each subclass have additional specific attributes which are simple attributes, parameters or variables, and which vary in number and type (Table 23). An inference-block is an object, preferentially used as an attribute of bioObjects, such as time-compartments, which represents an inference model, as defined by the methods providing the value of its Outcome, which is dependent on the values of some inputs. The inputs may be have any type of origins, including: a) one or more other attributes of the model-block; b) one or more attributes of other objects in the Virtual Model; c) one or more values originating from any external source interfaced with the system, such as external sensors, databases, or external simulators; d) any auxiliary independent constant, parameter, or variable; e) any timer or meter; or f) any combination of the above.

- **Class button**

- The **class button** (Table 24) is a subclass of bioTool, which additional attributes include Toggle-state. A subclass of button is action-button (Table 24), which additional attribute, action-proc-name, holds the name of the specific procedure invoked when that button is selected by the user. Clicking an action-button triggers its "select" option (Table 25) which invokes the button-handler-callback, which in turn invokes the procedure defined as attribute of each subclass, such as the one example shown for hide-sup-ws-callback (Table 26) for a HIDE button. Action-buttons play a supportive role in this invention by controlling the display, and in some

cases the activation, of workspaces, allowing the interactive navigation through the entire Virtual Model. Other subclasses of button, named tracers, initiate important interactive tasks to create and display pathways, lists, or graphs from a point of view within the Virtual Model relative to the bioReservoir or bioProcess from which they are selected.

- The **class change-mode-button** (Table 27) is a subclass of action-button with the additional attribute Mode, which upon selection starts the change-mode-callback (Table 27). A set of these buttons on a panel, with the Mode and Label attributes of each button of the set equal to each of the possible values of the user-mode attribute of the window, allows the user to interactively change from one user mode to another, for the particular window displaying the panel. The optional behaviors of many types of objects in this system are user mode specific, as defined within the Class-restrictions attribute of the object-definitions of the corresponding classes, such as the actions invoked when selecting their icons (Tables 20, 24, 28, 30, 32, 34, 47, 52, 53, 54, 65, 66), the options that appear on their menus, or the visible attributes.
- The **class path-tracer** (Table 28), a subclass of the class button with no additional attributes has subclasses with specific methods associated with them. The "show" option starts the path-tracer-handler-proc (Table 29), which calls the configure-tracer-proc (Table 29) and one of three optional procedures, depending on the class of the path-tracer from which this procedure was invoked, as described in later sections related to bioProcesses, Navigation Panel, Simulation Panel, and Experiment Panel, where they play a support role by controlling the dynamic creation and display at run-time of interactive pathways of either bioReservoirs or bioProcesses or both.
- The **class lists-tracer** (Table 30), a subclass of the class button with no additional attributes has subclasses with specific methods associated with them. The "show" option starts the list-tracer-handler-proc (Table 31), which calls the configure-tracer-proc (Table 29) and one of three optional procedures, depending on the class of the lists-tracer from which this procedure was invoked, as described in later sections related to bioReservoirs and bioProcesses, where they play a support role by controlling the dynamic creation and display at run-time of lists of either bioReservoirs or bioProcesses that are either upstream or downstream of the bioReservoir or bioProcess where the lists-tracer is located.
- The **class graph-tracer** (Table 32), a subclass of the class button with no additional attributes has subclasses with specific methods associated with them. The "show-plot" option starts the transf-graph-tracer-handler-proc (Table 33), which calls the configure-graph-tracer-proc and one of three optional procedures, depending on the class of the graph-tracer from which this procedure was invoked, as described in later sections related to bioReservoirs and bioProcess, where they play a support role by controlling the dynamic creation and display at run-time of graphs that plot the values over time of key variables or parameters of those bioObjects. The actions performed by these procedures are similar, creating first a display with a graph and each a different set of predefined complex variables and then establishing relationships among those variables and the graph. The components of such display are discussed in more detail below.

• **Other subclasses of bioTool** used in the current implementation of this invention include:

- The instances of the **class reference-block** (Table 34) are used to represent each reference to published information that serve as the source for building the different parts of the models. Selecting a reference-block displays the attribute table of each show information about the sources, such as the short-reference, authors, title, journal, institution and abstract.
- The instances of the **class bioObject-title** (Table 35) are used to label subworkspaces, such as those of bioEntities, bioEntity-notes, subModels. A set of rules (such as the examples in Table 80) automatically set the text of those titles, based on the label of the object superior to such subworkspaces.
- The classes **reference-message** and **sequence-display** (Table 36)are subclasses of the Shell's class message used for display of text in the References Scroll and the Mol.Sequence, described below.

e) **Group of Operating/Display Tools**

• The classes query-panel and entry-panel (Table 37) are both subclasses of the class uil-tailored-dialog, as defined in C2 Version 3.0, from which they inherit several attributes used for run-time processing. The different types of query-panels and entry-panels are designed as partially built master structures which are dynamically cloned and completed at run-time, upon requests from the user. The dynamically created Entry Panels (FIGs. 32, 33 and 35) are used to interactively set-up and start constrained navigations, simulations, and experiments, and for selection of monitoring/control components. The dynamically created Query Panels (FIG.30) are used to interactively perform predefined complex queries, based on functional modular structure, relative position within the pathways, function (roles as bioReactants), or location in subcellular compartments.

- Scroll-areas are structures provided by the Shell. In this invention Scroll-areas are created dynamically and used extensively (FIGs.30 through 35) to list and provide direct access named instances of different types of bioObjects. For example, a scroll-area is created upon request from an Experiment Panel (FIG.35) by selecting the BIORESERVOIRS button (3506), listing in a scrollable fashion all named instances of bioReservoirs in the Virtual Model and providing direct access to them. The scroll-messages that form the scrollable components of the scroll-area, and which refer to the objects of interest, can be ordered alphabetically or in other ways.

f) **Connections and bioPosts**

- **Connections** are used to connect visualObjects upon a workspace and are established graphically, by the union of the stubs attached to the bioObjects involved, or by extending a stub of a bioObject and attaching it directly to another bioObject, if the latter allows manual connections. Several domain-specific connections are subclasses of the Shell's class connection, and represent different abstract concepts such as inputs, outputs, links, relationships, or information passed between objects. Stubs are handles on the icons of objects that are clicked and dragged to create the connections. The definitions of the subclasses of bioEntity, bioPool, bioEngine, bioReactant, bioProduct and bioPost prescribe the class of connections that are allowed at each port of their icons, as defined in their stubs slot. Direction of flow may also be specified in such definitions. Stubs are color and pattern coded to represent the different types. The stubs are inherited, and those stubs that are not used can be graphically removed.

- The definitions of connection subclasses, some of which are shown in Table 38, include the descriptions of the **Cross section pattern**, that provide each with characteristic pattern and color to facilitate visualization and modeling tasks, and a predefined **Stub length**. Each of those classes is specific for connecting particular combinations of objects, as described below (FIGs. 6 and 7). Stubs of different classes or subclasses are not allowed to connect to each other, to prevent the modeler from making erroneous connections. Connections comprise the following major classes:
  - the **class p-connection** represents a bioProcess input to a bioReservoir, and more specifically a bioProduct's input to a bioPool, in terms of both unidirectional material flow and data flow. Several stubs for p-connections are defined at the top (716) of the class bioPool, which may graphically connected (717) to the one stub defined at the bottom of each biopool-p-post. Also several stubs for p-connections are defined at the bottom (720) of the class BioEngine, which may graphically connected (609) to the one stub defined at the top of each bioProduct.
  - the **class r-connection** represents a bioReservoir's input to a bioProcess through a bioReactant, in terms of both unidirectional material flow and bidirectional data flow. An r-connection may connect (719) a biopool-r-post to the bottom of a bioPool, or may connect (608) a bioReactant to the top of a BioEngine. Several stubs for r-connections are defined at the bottom (718) of the class bioPool. In addition, the class r-connection has many subclasses which differ in their characteristic cross section color pattern, and each connects a different class of bioReactant to a matching stub of the same subclass defined at the top of a bioEngine. The definition of each subclass of bioReactant specifies within the stubs attribute the specific subclass of r-connection, and the definition of each subclass of bioEngine specifies within the stubs attribute a set of specific subclasses of r-connections and their locations. The purpose of this implementation is to prevent users from making the wrong connections, while directing the user to make the appropriate connections by matching the color pattern.
  - the **class model-box-connection** has three subclasses, scaled-input-box-connection (721), input-box-connection (722), and output-box-connection (723), used to graphically connect instances (FIG.8) of scaled-input-model-box (811), input-model-box (816), and output-model-box (817), respectively, to predefined corresponding ports of a bioPool. The stubs for these specific connections are defined for each subclass of model-box, and for each subclass of bioPool at named ports, and these subclasses are color coded to guide the modeler to match the color pattern when making the connections;
  - the **class link** defines the connections between the component bioEntities (3019, 3021) of other bioEntities, representing the continuity of an ordered sequence of structural components.
  - the **class icon-wire** is a tool used to connect a bioRole-post (703, 707, and 711) to its bioRole-Object. The **class graph-link** is a tool used to connect either a bioEngine to various types of tracers (3505, 3510, 3517, 3826) or a bioPool to various types of tracers (3821, 3828). After the connection is made, these connections are made invisible by either a rule at run-time or by a procedure invoked at initialization time ,

- The **class BioPost** (Table 39) is a subclass of of the Shell's class connection-post. Connection-posts are like extensions of connections that permit to physically interrupt a graphic connection and continue it on another workspace, but without interrupting the flow. A teaching of this invention is the design, uses, and methods associated with the different subclasses of bioPost, each referring to an iconic structure (703 or 711) that may be distantly connected to another bioPost of a complementary subclass (704 or 712, respectively). BioPosts are components located on the subworkspaces of bioReservoirs and bioProcesses and, since they are located upon different subworkspaces, they are distantly connected by receiving the same name. These structures integrate

bioReservoirs and bioProcesses, wherever they are located. There are several restrictions built into the system of this invention to prevent users from making the wrong connections, or to warn them that wrong connections have been made, such as when a bioReactant-post is given the same name as a bioPost that is not a biopool-r-post, or when a bioproduct-post is given the same name as a bioPost that is not a biopool-p-post. The various subclasses of bioPost (Table 1) will be further described below, within the overall description of bioReservoirs and bioProcesses.

g) **BioObjects**

- The class **bioObject** (bio-object, Table 40) is a subclass of the Shell's class object, with various subclasses and various restrictions when in different modes, as listed. BioObject comprises all the classes of knowledge structures used to represent the biochemical systems characteristic of this domain. These object classes are characterized by their defined attributes, which can be: a) simple attributes of various types, such as text, symbols, integers, floats or truth values; or b) other previously defined objects, including parameters and variables of any of those types, or lists and arrays of those types of elements. The values for the parameters and variables can be provided by any of a variety of sources, such as the user-inputs, interfaced on-line sensors, or dynamically by the inference engine (for instance, through specific or generic formulas, rules or procedures), or a simulator (for instance, through specific or generic simulation formulas or simulation procedures), reflecting the changing values of the attributes defined within the knowledge structures and in the context selected by the user. The current implementation of this invention integrates encapsulated mathematical models made of simulated variables and parameters which values are computed by the Shell's simulator.
- All bioObjects have at least three common attributes: a) **Names**, a Shell-defined attribute that binds one or more unique symbols to an object; b) **Label** a text attribute that allows to identify bioObjects that may not have a unique name; and c) **Description** is an optional text string that contains any additional information. While names are required in the current implementation for some bioObjects to be integrated into the pathways, such as bioReservoirs, bioProcesses, and bioPosts, labels are preferred for identification in other cases, because they require less memory than symbols, do not have to be unique, their syntax is less restrictive, and offer more attractive options for display.
- Among the major bioObject's subclasses, which further subclasses are described in later sections, are:
  - class **bioNode-Object** (Table 40): inherits all attributes and icon from bioObject and have no specific attributes, and can only have subclasses, not instances. However, there are methods that refer to this class, and therefore, like with many other classes throughout this system, the differentiation between classes may not appear in their definitions but rather on their behavior, through methods that applied to them.
  - class **bioView-Object** (Table 40): inherits all attributes and icon from bioObject, and can only have subclasses, not instances. A user-menu-choice labeled **Details** is defined for this class. Upon selection of this option from the icon of any instance of any subclass of bioView-Object by the user starts the procedure go-to-view-proc, which takes among its arguments the instance from which it was invoked and the current window and, depending on the value of the instance's **Toggle-state** attribute of hide or show, it causes the instance's subworkspace to be displayed or hidden, it toggles such value to the other value, which is an event that fires a rule that causes the instance's icon to be configured to its pressed or depressed appearance, and respectively. New attributes defined for this class comprise:
  - "**references**", defined for several subclasses of bioObject, is an optional text that makes reference to the sources of information and data used to construct and configure that instance of any subclass of bioObject for which it is defined. This attribute may be defined as: a) a simple text attribute, which the user may consult to perform independent searches, or using this invention search facilities, such as the Master-Reference-Panel (not shown); or b) a reference-array (Table 17) in combination with the user-menu-choice "show-references", which when selected by the user starts the show-references-of-array-proc (Table 107), which displays dynamically created copies of the instances of reference-block (Table 34) referred to by the values of that array.
  - "**warnings**" is an optional simple text attribute which value is set by the modeler to warn the user about any abnormalities or data included, such as, for a bioReservoir, when that particular pool of molecules have not yet been observed experimentally, but can be assumed to exist by analogy to similar systems, or when the pool has been observed but the quantitative data has been assumed. This attribute is used in conjunction with a design (723, 843) in the icons defined for all those classes and their subclasses as a separated color-region, called flag-color, that is changed to yellow to make the user aware of the existence of warnings, or back to any appropriate color. Those changes are automatically executed by procedures invoked during initialization, or at run-time by a set of rules.
  - "**toggle-state**" is an attribute also defined for objects which icons' appearance change upon selection by the user, depending on the value of this attribute which alternatively switches between show and hide when the user clicks on the icon.

h) **BioEntities**

- A bioEntity is an iconic knowledge structure that represents any biochemical entity or their components, at different levels of structural complexity. Each bioEntity may have information encoded in various ways, including: a) information and data stored in its table of attributes, and b) each bioEntity may have a subworkspace upon which the components that represent its functional modular structure are visually defined. As shown in FIG.30, selecting from a bioEntity icon's (3004) menu (3005) the "details" option (3006) displays its structure (3007), which may have several layers of detail (3008 through 3022). In the current implementation, functional components represent structural components that are relevant for the bioEntity's function and regulation, that allow the user to visualize the structural composition. The iconic composition is also used in the inference used for processing of queries that involve the structure, which refer to bioEntities that have certain components in their subworkspace. Each functional component of a bioEntity belongs by itself to a bioEntity class, either molecule or bioEntity-component or any of its hierarchy of subclasses.
- The **class bioEntity** (Table 42), a subclass of bioObject, has numerous subclasses, organized in a hierarchical structure with different levels, according to the way a biochemist would classify different molecules, by taking into consideration their commonalities in chemical composition in the upper levels of the hierarchy, such as protein, nucleic acid, or lipid, followed by a reference to their function, such as, within the class protein, active-polypeptide, receptor, or enzyme. Table 41 shows a partial listing of such hierarchy, with some examples of the subclasses at different levels. Additional attributes include **References**, described above.
- There are two major subclasses of bioEntity:
- The **class simple-bioEntity** (Table 43) includes all the subclasses with simple structures that are not further visually described in the current implementation, and they do not generally have subworkspaces with additional components. This class has two subclasses (Table 43), protein-site, and protein-modified-group, each with its additional subclasses: Additional attributes include:
- **Id** is a simple attribute that is used to hold a text to identify the instance instead of using a unique name;
- **Position** is a simple attribute text that indicates the position of that component within the molecule when applicable, such as the position of an aminoacid in the sequence of a protein, and is usually displayed; and
- **Superior-bioentity** is a simple indexed attribute that holds the name of the first superior bioEntity in the workspace hierarchy that has a name, if any, a value that is computed by the program within the query reasoning and, as explained later, this mechanism is provided to avoid repetition of structures that are shared by various instances of the same family or of other families that share regions of homology.
- The **class complex-bioEntity** (Table 44) includes all those subclasses that can be further described by their components in their subworkspaces. As shown in FIG.30, selecting from icon's (3004) menu (3005) the "details" option (3006) displays its structure (3007), which may have several layers of detail (3008 through 3022). Additional not previously described attributes include:
- **Master-bioentity** is a simple indexed attribute that may hold the name of a bioEntity that is displayed when the user selects the "details" option. This value is set by the program upon cloning an existing named bioEntity using the create-local menu option, as described later. BioEntities with no names which Master-bioentity names another bioEntity, the master bioEntity, can be considered as only a copy or pointer to the subworkspace, or **Master-details**, of the master bioEntity, but it can also hold additional information in its table of attributes or in its own subworkspace, or **Local-details**.
- The major subclasses of complex-bioEntity, defined to deal with multiple levels of biochemical structural complexity, include:
- the **class bioEntity-component** and its subclasses (Table 44) represent he lower-level molecular functional components such as: dna-component and its subclasses, such as dna-domain, promoter, operator, and so on; protein-component and its subclasses, such as protein-domain, protein-motif, and so on; and membrane and its subclasses. The subclasses of this class may have 4 stubs of the class link used to link different bioEntity-components, allowing for branching, and those stubs that are not used can be dragged away. Additional attributes include: **Superior-bioentity** and **Position**, defined above; and **Size**.
- the **class molecule** (Table 45) and its successive level of subclasses such as those of **protein** (Table 45), **nucleic-acid**, and so on, represent the next level of complexity. For example, as shown in FIG.30, a protein may have a set of domains or motifs, or a combination of both, and it may have in addition one or more protein-sites, or protein-modified-groups, connected to the domains and or motifs at approximate locations, with the more accurate position displayed in the "position" attribute. A protein-domain may also have in its SW a set of smaller domains or motifs, or a combination of both, and it may have in addition one or more protein-sites, or protein-modified-groups, including copies of those connected to the icon of the domain.
- the **class heter-mol-complex** and its subclasses, such as nucleosome, tf-dna-complex, represent a higher level of complexity. A heter-mol-complex or a complex-molecule may be composed of other molecules. For example, a

t-dna-complex may have a dna or a gene icon in association with one or more icons of specific transcription-factors. In turn, the structure of a simple molecule is represented by an ordered set of bioEntities of the class bioEntity-component in its subworkspace.

- the class **membrane** and its subclasses, are used to represent interactions of several types of cellular membranes with other molecules;

- In addition to the structure in its subworkspace, additional information about an instance of bioEntity can be stored in its table of attributes, such as the example shown for a default instance of the subclass p.tyr.kinase (Table 45). The inherited attributes shown are successively added through the hierarchy of its superior classes molecule, protein, and enzyme, and include:

- **Synonyms** is a simple text attribute to list the different names by which a molecule or complex may be known;
- **In-species** is a simple text attribute to list the species or common name of living organisms in which that molecule or complex has been found, such as all, human, mouse, Drosophila, C. elegans, S. cerevisiae, and so on;
- **In-tissues** is a simple text attribute to list the tissues in which that molecule or complex has been found.
- **InCells** is a simple text attribute to list the cell types in which that molecule or complex has been found.
- **InOrganelles** is a simple text attribute to list the organelles within a cell in which that molecule or complex has been found, and it may take values such as plasma-membrane, cytosol, nucleus, and so on.
- **Cas-number** is a simple text attribute to indicate the CAS number assigned to that molecule, which is indexed to allow for faster searches and has a default value to indicate the preferred format;
- **Mol-weight** is a simple integer attribute to indicate the molecular weight of a single molecule or complex, preferably expresses in Daltons;
- **Isoelectric-point** is a simple float attribute to indicate the isoelectric point of the molecule or complex;
- **Sequence** is a simple text attribute to contain information about the sequence of a molecule, if relevant;
- **Substrates-info** is a simple text attribute to contain information about the known specific substrates or groups targeted by the enzyme;
- **Inhibitors-info** is a simple text attribute to contain information about the known specific inhibitors of the enzyme;
- **Ligands-info** is a simple text attribute to contain optional information about any other ligand that may bind to the enzyme;

- The **class bioEntity-notes** (Table 46), a class of bioTool, comprise auxiliary structures located the subworkspace of a bioEntity, that serve as iconic containers for different types of specific information about the bioEntity, which can be displayed by selecting the "details" menu option.

i) **BioReservoirs**

- **Class bioReservoir**

- As shown in FIGs. 12 and 13, a bioReservoir (1201) is an iconic object that represents an imaginary container of a population of similar molecules or molecular complexes, represented by a bioPool (1301). A bioReservoir has several operational functions and encapsulates several forms of knowledge. A bioReservoir's menu (1202) provides access to a variety of tasks to be discussed in sections below. Here we will define some of the components involved. The "table" option (1203) displays its table of attributes (1204), which holds values set by the modeler that, in addition to provide useful quantitative or qualitative information to the user, may be used by the program to set the initial conditions before a simulation is run. The values of some of the attributes of a bioReservoir characterize the system and are stored as part of the permanent database. The values of other parameters and variables pertaining to a bioReservoir which are computed when a simulation is run, but which do not usually require to be set by the modeler, are hidden for convenience as attributes of the bioPool contained in each bioReservoir, as described below, but they could as well be implemented as attributes of the bioReservoir.

- The **class bioReservoir** (Table 47) is a subclass of bioView-Object and its subclasses (Tables 47 and 48) include: bound-mol-reservoir, sol-mol-reservoir, cell-reservoir, exp-cell-reservoir, genet-mol-Reservoir, some with further subclasses . All instances of all subclasses of bioReservoir have an activatable-subworkspace, which means that any object upon such subworkspace are not recognized by the inference engine or the simulator unless the subworkspace is activated, and therefore any of the tasks associated with their icons are not functional when inactivated. When the application is started, all those subworkspaces are deactivated, and in the current implementation of this invention they are activated when the general initialization procedures are invoked, as described below. In FIG.12 is shown a table of attributes (1204) of an instance not yet configured, shown here with its default values. The unlikely default value of 9.9e-99 given to some of those attributes, which provides a) a signal for the user that an adequate value has not been entered yet by the modeler, and b) a branching criteria for the inference engine. Attributes newly defined for this class and its subclasses not previously described include:

- **Compartment** (1205) is an simple attribute optionally set by the modeler to represent the physical boundaries of the contained bioPool;
- **Status** (1206) is a symbolic attribute which value is set and used at run time by the simulation and other procedures, as an aid in the activation of pathways and in the creation of interactive pathway displays. It can take any of the values specified in the definition table and it is not visible to the modeler or other users.
- **Ref-bioentity** (1207) is an optional attribute optionally set by the modeler to point to the bioEntity instance that describes the structure of a unit representative of those populating that bioReservoir. This value is used by the program when the user selects the "bioEntity" option from the menus of either the bioReservoir itself or any of the bioReactants and bioProducts distantly connected to its bioPool, and that option appears on those menus only when the value of this attribute is not the default value none;
- **Master-bioreservoir** (1208) is set only by the program when copies of named bioReservoirs are made, such as those used in the creation of interactive pathway displays, and is used by the program to display the subworkspace of the bioReservoir referred to by the value of this attribute, when the "master-details" menu option is selected by the user. This attribute is not visible to the modeler or other users.
- **Decay-rate-factor** (1209) is a parameter which value may be entered by the modeler or can be modified at run time by rules or procedures. Its default value is 2.0e-4, a value small enough not to cause dramatic changes, but in line with some experimental observations in the domain of this invention, but it should preferably be modified by the modeler to better reflect individual cases;
- **If-scaling-amount** (1210) is a simple attribute set by the modeler to indicate to the program the value to be used to interconvert between the absolute-valued and the scaled-valued variables or parameters of bioPools and their connected bioReactants. The complex methods used for this conversions differ, depending on the classes of bioPools and bioReactants involved, as discussed in more detail under the Simulation Mode heading. Its default value of 100 is used if not modified by the modeler, if the value of the if-scaling-bioreservoir attribute is none;
- **If-scaling-bioreservoir** (1211) is a simple attribute which value may optionally be set by the modeler to indicate to the program that the value of the if-scaling-amount should be taken from the bioReservoir named by this attribute. This value is only required or used when the value of the if-scaling-amount attribute is the default, otherwise, the later value is used and this attribute is ignored. This and the previous attribute are not visible in General Mode;
- **Scaled-basal-amount** (1212) is a float simple attribute set be the modeler that represents the scaled value of the amount of units in the contained bioPool under normal or basal conditions, equivalent to a fraction of the maximum amount that this bioPool can reach under optimal physiological conditions. Since in many occasions in the domain of this invention neither the basal nor the maximum amounts can be measured or known with certainty, this value represents in such occasions the best estimated guess, and it is an important semi-quantitative knowledge component. The default is the unlikely value of 9.9e-99.
- **Physiol-abundance** (1213) is a symbolic attribute which value is set by the modeler to indicate the physiological level of abundance of the contained entity in the compartment represented by that bioReservoir. It is a symbolic replacement of the previous attribute when the 9.9e-99 has not been modified by the modeler. The values allowed for this attribute in this invention are: highest, abundant, steady-state, low-induced or only induced, which currently correspond to the scaled values of 1.0, 0.9, 0.5, 0.1, and 1.0e-9, respectively. However, this default scale can be changed to different values, and additional symbols and values can be added to the scale. As and alternative, fuzzy-sets can be defined for the values of this attribute, and fuzzy-logic can then be applied to those values. The reasoning for using the values of either the previous or this attribute, in conjunction with one or more of the following attributes, are discussed in more detail under the Simulation Mode heading.
- **Normal-basal-concentration** (1212) is a float simple attribute set be the modeler that represents the physiological average value of the concentration of the entity contained in a sol-mol-reservoir;
- **Normal-basal-density** (instead of 1214) is a float simple attribute set be the modeler that represents the average value of the density of the entity contained in a bound-mol-reservoir.
- **Physiol-max-concentration** (1215) is a float simple attribute set be the modeler that represents the physiological maximum value of the concentration of the entity contained in a sol-mol-reservoir;
- **Physiol-max-density** (instead of 1215) is a float simple attribute set be the modeler that represents the physiological maximum value of the density of the entity contained in a bound-mol-reservoir.
- Selecting the "details" option (1216) displays the subworkspace (1217) which contain the components that characterize a bioReservoir. The classes of those components that are using in the modeling process are defined below, while other auxiliary structures are described in later sections. The descriptive qualitative, structural, and functional information of a bioEntity is kept in this invention separated from the configurable additional quantitative information that characterizes each bioReservoir, allowing the Ref-bioentity of several

bioReservoirs, representing populations of the same type of entity in different locations or points in time, to point to the same bioEntity to reduce the size of the Virtual Model.

- **Class bioPool**

- The class bioPool (Table 49) is a subclass of bionode-object and comprises several subclasses, such as sol-mol-pool, bound-mol-pool, ... , including those defined in Tables 50 and 51. A bioPool is defined with a characteristic icon with two sets of stubs of two classes of connections: a set of p-connection stubs at the top of the icon, and a set of r-connection stubs the bottom of the icon. As described above, each p-connection stub (716) is to be connected (717) to a biopool-p-post (712) and each r-connection stub (718) is to be connected (719) to a biopool-r-post (704) on the subworkspace of the bioReservoir, which allow to establish distant connections to a bioReactant and a bioProduct, respectively, and allow bidirectional flow of data and control between them. As shown in FIG.13, every instance of bioPool (1301) has an associated menu (1302), and selecting the "table" option (1303) displays its table of attributes (1304). Several of the attributes of a bioPool are variables or parameters, which include a density-related set and a concentration-related set, which values do not have to be set by the modeler (with the exception of the basal quantities, which are preferably set by the modeler if known, but they are not required, since they can be computed from the set values of other attributes of the bioReservoir, or their default values), since they are inferred or simulated, as described under the Simulation Mode heading. The units of all the parameters and variables of all bioPools connected to the same bioProcess have to be appropriately matched. Newly defined attributes of bioPool comprise:
- **Basal-Density** (1305) is given by a basal-density-par (1306). The subclass sol-mol-pool has additionally a **Basal-Concentration** (1321) given by a basal-conc-par (1322). The values of either of these attributes may be set by the modeler to indicate: a) the normal physiological steady state density or concentration, or any other appropriated measured or assumed value, for physiological molecules or complexes, or b) 0.0 for pharmacological or other molecules that are added to the system from an external environment. This value is required for any bioReservoir that participates in a simulation, and if its default value has not been overridden by the user with a new value, the program sequentially pursues other sources of the Basal-Density value, as determined by a generic basal-density-procedure which is called during the activation process of a simulation, only for those bioReservoirs that have been activated for a simulation;
- **Scaling-Density** (1307) is given by a scaling-density-par (1308) which value is inferred by a procedure invoked at initialization, as discussed in more detail under the Simulation Mode heading. This value is used by simulation formulas to interconvert scaled values and absolute values.
- **Density-Entry** (1309) is given by a density-entry-pvar (1310). It takes the value of a user-input at a time, or different times intervals, during simulation as defined by the user in an input-panel, during the simulation set-up process. It has a default value of 0.0, and a new value may be inferred and set by a procedure called when the simulation is started, and, after the new inferred value has been propagated, it reverts back to the default value of 0.0. This value is an argument to the Accumulation, where it is summed to other inputs and outputs;
- **Scaled-Entry** (1311) is given by a scaled-entry-pvar (1312), and it is the scaled equivalent of the Density-Entry, with its value set and used in a way similar to that described above;
- **Input-Rate** (1313) is given by a input-rate-pvar (1314), which value is given by a simulation formula that sums all the visually defined inputs, including: a) the current values of the Production-Rate of all the bioProducts connected to the biopool-p-posts of the bioPool, if any; and b) the input modeled by means of a model-block contained in any connected input-model-box (or scaled-input-model-box), if any;
- **Output-Rate** (1315) is similarly given by a output-rate-pvar, which value is given by a simulation formula that sums all the visually defined outputs, including: a) the current values of the Consumption-Rate of the consuming bioReactants connected to the biopool-r-posts of the bioPool, if any; and b) the output modeled by means of a model-block contained in any connected output-model-box, if any;
- **Accumulation** (1316) is given by an accumulation-pvar, a time variant and continuous state variable which represents a quantity in the classical systems dynamics sense which, given an initial value here set to be equal to the basal-amount (or scaled-basal-amount), as discussed under the Simulation Mode heading, integrates the input-rate, the output-rate, the input entered by the user through an input-panel, if any, and the decay term, which is a function of the current value of the Concentration or Density as given by the decay-rate-factor of the bioReservoir (a decay rate constant in sec^(-1)) which represents a variety of outputs not modeled visually or through a model-block, including the degradation and diffusion components). Since at steady-state the overall rate of output is equal to the overall rate of input, then, by definition, the Accumulation should approach 0.0 in a closed system, in the absence of additional external disturbances, such as user-inputs or inputs modeled by model-blocks.
- **Density** (1317) is given by a density-pvar (1318) which value represents the number of molecules, complexes, or cells per liter, or any other unit of volume, such as the volume of the reaction mixture in a reactor. The subclass sol-mol-pool has the additional attribute **Concentration** (1317) given by a scaling-density-par (1318), which

value in M (Molar) represents moles per liter. In the implementation used to illustrate this invention, simulations operate by default based on the Density of the bioPools involved. In the case of sol-mol-pool, where the quantities used by scientists may be most commonly given as Concentrations, the second set of concentration related variables is defined to facilitate user-input, but those quantities are preferably transformed into density-related variables by the program, using Avogadro's number (6.023e23 molecules per mol), before integration with other quantities. However, several other types of quantities could be used as well, with little conversions to other units of measure, such as activity (X-units times Y-unit), amount (X-units), or others types of densities (such as X-units per Y-unit). The same concepts and the same type of formulas, rules and procedures, apply to other-quantities, and a variety of units may also be allowed for entries from the user or other external sources, by providing lookup-tables or any other standard methods for automatically converting any of those units to the desired target units. Hence, the modeler has the option to design and define a system totally or partially based on a different quantity measurement, as long as proper care is taken to maintain consistency throughout the system. In those cases, the quantities entered by the modeler in the attribute slots of the bioReservoir that refer to physiological levels would also have to reflect those different sets of units.

- **Scaled-Amount** (1319) is given by a scaling-density-par (1320), which value represents an scaled, dimensionless, alternative to Density, Concentration, or any other absolute-valued quantity. This attribute is used in the default scaled reasoning, which is the preferred method when dealing with biological systems, when absolute quantitative data is incomplete, and much of the data available is from relative measurements. The Scaled-Amount can be converted into a Density, and viceversa, by using the value of the Scaling-Density of the bioPool.

• **Class bioPool-post**

- The **class bioPool-post** is a subclass of bioPost (Table 52) and has two subclasses: biopool-p-post (712), which is to be connected to an input p-connection at the top of a bioPool's icon, and its name is to be given to one bioproduct-post to establish a distant connection; and biopool-r-post (704, 708) which is to be connected to an output r-connection at the bottom of a bioPool's icon, and its name is to be given to one bioReactant-post to establish a distant connection. BioPool-posts are located upon the subworkspace of a bioReservoir and connected a bioPool, which function in combination with bioRole-posts is to connect the bioPool to one or more bioEngines. The additional attributes for this classes include:
- **Id** is a simple text attribute used as a unique identifier, instead of the unique name, in some types of processing, such as in the alternative simulation procedures described in that section;
- **Ref-bioprocess** is a pointer to the bioProcess instance that contains the bioProduct or bioReactant to which the biopool-p-post or biopool-r-post, respectively, are distantly connected. The value is interactively set by selecting the "set-refs" menu option of the bioproduct-post or bioReactant-post to which it is distantly connected;, and it is used by the program for interactive navigation through bioReservoirs and bioProcesses, as described below.

• **Class model-box**

- As shown in FIG.12, a model-box is an iconic object located upon the subworkspace of a bioReservoir and connected to its bioPool, which function is to hold and connect the optional model-blocks on its subworkspace to the bioPool. The class model-box (Table 53) is a subclass of bioTool and has three subclasses, each playing a different role as their name indicate: scaled-input-model-box (1218), input-model-box (1236), and output-model-box (1237). Model-boxes are interactively created by selecting a menu option of the closest bioPool, as described below. An instance of model-block is connected manually to the connection-post (1238) which comes with the subworkspace of a model-box. The superior connection of such connection-post is that defined by the stub of the model-box that contains it, and therefore, the model-block (1222) connected to it is distantly connected to the bioPool to which the model-box (1218) is connected. The menu options available for all subclasses of model-box are different, depending on the user mode. When clicking on a model-box (1218) in any of the user modes bypasses its menu (1219) and selects the "show-sw" option (1220), which causes its subworkspace (1221) to be displayed.

j) **BioProcesses**

• **Class bioProcess**

- The class bioProcess (Table 54) is a subclass of bioView-Object, and comprises the subclass cell-bioProcess. As shown in FIGs. 14 and 15, a bioProcess is an iconic object (1401) which subworkspace (1407) contains components that visually represent the preconditions (bioReactants), the process engine (bioEngine) and the effects (bioProducts) of a process as connected objects with encapsulated variables and parameters that describe the process qualitatively and quantitatively. The subworkspace of bioProcesses is activatable, allowing to control the availability of its components by activating or deactivating the subworkspace. In the

current implementation those subworkspaces are activated with the general initialization procedures, as described under the General Mode heading. The many different types of bioProcesses are all instances of that class or its subclass, and differ mainly in the iconic components upon their subworkspace. The different types are also visually distinguished by different colors for the type-color region of their icon, which is changed programmatically to be the same color as the color of the type-color region of the bioEngine they contain.

- A bioProcess has several operational functions and encapsulates several forms of knowledge. Each instance of bioProcess has a menu (1402)provides access to a variety of tasks to be discussed in sections below. Here we will define some of the components involved. The "table" option (1403) displays its table of attributes (1404) which holds various attributes, some of which have values that are set by the modeler and provide information for the user, and others are auxiliary attributes hidden from modelers and users which values are set and used only by the program at run-time. Most attributes specific for this class have been previously described for other classes. Newly described attributes include:
- **Master-bioprocess** (1405) is set by the program when copies of a named bioProcesses are made, such as those used in the creation of interactive pathway displays, and is used by the program to display the subworkspace of the bioProcess referred to by the value of this attribute, when the "master-details" menu option (visible only when the attribute has a non-default value) is selected by the user. This attribute is not visible to the modeler or other users.
- The "details" option (1406) displays the subworkspace (1407) with all its visual components. In addition to other auxiliary structures to be discussed below, the main components comprise a single bioEngine (1501) which represents the interactions between the inputs represented by the bioReactants (1408 and 1419) where the bioProduct(s) (1518) are generated.
- As described below (FIG.32), a cell-bioProcess may contain bioReactants and bioProduct that represent pools of cells, in addition to containing bioReactants and bioProduct that represent pools of chemicals. So it is possible to have bioProcesses where molecules interact with molecules, as well as cell-bioProcesses where molecules (3205) interact with cells (3204), or cells interact with cells. In addition, there may be a pool of pairs of interacting-cells as a result of cells interacting of cells (not shown). Other type of bioProcesses represent translocation processes between location compartments, where the bioReactant represents a fraction of a given bioPool in a given compartment which is removed from that bioPool and transferred, with a time-lag if so desired, to the target bioPool which is connected to the bioProduct, and which represents a different pool the same entity in a different compartment. Additional types of bioProcesses represent transfer of cells from a bioPool representing one state of those cells to another bioPool representing a different state of those cells. Each of these types of bioProcesses contain the appropriate subclasses on bioEngines. There are many other types of bioProcesses defined in this invention and, in addition to those examples already mentioned, some others are discussed with the modeling approaches below. Many more types of processes can be represented by composing additional types of bioProcesses by a modeler skilled in the art.

- **Class bioEngine**

- The **class bioEngine** (Table 56) is a subclass of bionode-object and comprises a hierarchy of subclasses. A few examples of which incremental definitions are shown in Tables 56 through 58, such as: amplifier-bioengine, binding-bioengine, lumped-bioengine and cell-bioEngine. Each bioEngine's icon has a number of stubs of different types of r-connections at the top, to be connected only to bioReactants, and a number of stubs of p-connections at the bottom, to be connected only to bioProducts. The number and class of defined specific r-connections specify the different classes of bioReactants to be connected, such as enzyme.r, substrate.r, inhibitor.r, receptor.r, agonist.r, antagonist.r or unit.r. The bioEngines of the class lumped-bioengine, including those with two or more enzymes, represent lumped models of a two or more reactions in series, and are used to model stretches of pathways which detail is either not desired or not known, and only the rate limiting reactions are represented.
- The major subclasses of bioEngine (Table 55) are arranged according to the type of reaction or process they represent, indicated by their names. The next level of classes usually refers to the type of activity of the members of the class and is usually characterized by the class of the variable that defines its Velocity attribute. Further subclasses of each of those classes are defined according to the number and type of their defined stubs, which in turn determines the number and classes of connected bioReactants, and further by color coding the icons to show further specialization, such as using different type-color for different types of enzymes, such as protein-kinases, protein phosphatases, and so on. Further differentiation between particular instances can be generated by selecting as the value of their Velocity attribute an instance of different subclasses of velocity-pvar. The nomenclature used for each subclass refers to the defined bioReactants, such as:
- bioengine.E1.S2.I1.Em is an enzyme-bioengine that represents an enzymatic reaction and which icon has at the top four stubs: one for the enzyme.r, two for substrate1.r and substrate2.r, and one for an inhibitor.r.

- bioengine.R1.L1.An1.Rm is a receptor-bioengine which icon has at the top three stubs: one for the receptor.r, one for ligand.r, and one for an antagonist.r;
- bioengine.M1.Ef1.An1.CCm is a conform-change-bioengine that represents an induced conformational change process which icon has at the top three stubs: one for modifier.r, one for effector.r, and one for an antagonist.r;
- bioengine.U2.CFm is a complex-formation-bioengine that represents the formation of a complex from two previously independent bioEntities (which may themselves be complexes) and which icon has at the top two stubs: one for unit1.r and one for unit2.r;
- bioengine.C1.M1.CDm is a complex-dissoc-bioengine that represents the dissociation of one complex induced by a mediator into two or more independent bioEntities (which may themselves be complexes) and which icon has at the top two stubs for complex.r and mediator.r.
- bioengine.ch1.ii1.L1.ICm is a channel-bioengine which icon has at the top three stubs: one for channel.r, one for ion-Input.r, and one for antagonist.r;
- bioengine.C1.In1.Tm is a translocation-bioengine that represents a translocation between different compartments and which icon has at the top two stubs: one for carrier.r and one for input.r;

+ As shown in FIG.15, a bioEngine (1501) is located upon the subworkspace of a bioProcess that represents the action of any process, such as synthesis, modification, complex-formation, translocation, diffusion, degradation, and so on. Selecting the "table" option (1503) from the menu (1502) of a bioEngine (1501) displays its table of attributes (1504), in this case with the default values. Newly defined attributes not previously described include:

- **Velocity** (1505) is an attribute specific for each subclass which value is an instance of any of the subclasses of rate-pvar, described above, which for the receptor-bioengine shown is a binding-rate-pvar (1506). The values for this dependent variable are provided by generic simulation formulas that model the rate of the interactions of the bioReactants connected to the bioEngine. Its arguments may be the Contributions of each of its bioReactants, when using the scaled set of variables and formulas, or the bioReactant's kinetic-coefficients and the Densities, Concentrations or other quantities of the bioPools connected to the bioEngine through the bioReactants. Its output is an argument for the Consumption-Rate and Production-Rate of the connected bioReactants and bioProducts, respectively. Therefore, this attribute may take as arguments either a set of scaled-valued variables and/or parameters, in which this attribute would be also scaled-valued, or a set of absolute-valued variables and/or parameters, in which this attribute would be also absolute-valued, respectively.
- **Rate-constant-sec** (1507) is a simple float attribute which value may optionally be set by the modeler to provide the rate-constant in seconds for the overall process. This value is used, in combination with the Contribution, by the set of generic simulation formulas that use the scaled set of variables, which is currently the default mode of operation. The default value is used if no other value is set by the modeler.
- **Tau-coeff** (1508) is by default a simple float attribute which value may optionally be set by the modeler to modify the rate-constant-sec by such factor, allowing for testing the effects of a modification of such parameter on the simulation of the system without having to change the value of such parameter which may have been obtained experimentally. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, multiplying by its default value of 1.0 has no effect on the system.
- **Bias** (1509) is a simple float attribute which value may optionally be set by the modeler to modify by a constant amount the rate-constant-sec, for testing purposes, without having to change the value of the later attribute. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, adding its default value of 0.0 has no effect on the system.
- **Time-lag** (1510) is an optional time-interval attribute that may be useful for certain instances of bioEngines, which value, given by a time-delay-par (1511), is optionally set by the modeler to indicate the period of time by which the forwarding of the computed output of the Velocity of a bioEngine is delayed. It represents a time delay, equivalent to holding an amount from a bioPool for the specified period of time, which is returned after that time has elapsed. An example of its use is when a bioReactant and a bioProduct of this process are both connected to the same bioPool, and the forwarding of the output value of the Velocity is delayed. The value of this attribute may also be modified at run time, as a result of dynamic events related to this or other processes, and may be simulated or inferred by means of modeler-defined formulas, rules, or procedures;
- **pH** (1512) and **Temperature** (1516) are optional float attributes which values are given by a ph-par (1513) and a temperature-par and which, as their name indicate, represent the environmental conditions under which the process takes place. The values of these parameters may vary at run time, as a result of dynamic events related to this or other processes, and may be simulated or inferred by means of modeler-defined formulas, rules, or procedures;
- **pH-deviation-factor** (1514) and **Temp-deviation-factor** (1517) are optional float attributes which values are given by a ph-factor-par (1515) and a temp-factor-par and which are used in conjunction with "pH" or

"temperature", respectively, and represent a correction factor that either accelerates or deaccelerates the process when the values of either the pH or the temperature deviates from their defined default values. If used, the values of these parameters may remain constant or may vary for different values of the pH or temperature, as defined by a tabular function similar to a look-up table.

- **Class bioRole-Object**

- The **class bioRole-Object** (Table 59) is a subclass of bioObject, and has two subclasses: bioReactant and bioProduct. A bioRole-Object is an iconic object that represents the role that units from a bioPool plays in a bioProcess, either as a different types of bioReactants, such as: enzyme.r, substrate.r, inhibitor.r, subunit.r, receptor.r, agonist.r, antagonist.r, carrier.r, and so on, each representing the specific roles of the inputs that different bioReactants provide to the bioEngine, or as a bioProduct.

- As shown in FIG.14, a bioReactant (1408 and 1419), is connected by an input r-connection at the top of a bioEngine, and represents the material contribution from a bioPool to the bioEngine to which it is connected. To be operational, the bioReactant-post connected to a bioReactant must have the same name as one of the bioPool-r-posts connected at the bottom of a bioPool. The two bioReactants shown are representative examples of a pair of two interacting molecules or complexes with complementary roles or functions: a receptor and its ligand. As described earlier, may other combinations of two, three or any other number of interacting participants are possible, to represent other types of processes, such as combinations of enzyme and substrate(s), units that form a complex, and so on. These combinations may also include antagonists or inhibitors, which may alternatively be represented as competing in a separate process, the latter being the preferred alternative in the default implementation using the scaled set of variables and simulation formulas.

- The **class bioReactant** (Table 61) comprises a hierarchy of subclasses, such as: amplifier-bioreactant, source-bioreactant, leading-bioreactant, binding-bioreactant, inhibitor-bioreactant, single-bioreactant, cellReactant, cell-receptor, extracell-ligand, and their subclasses. A few examples of their incremental definitions are shown in Tables 61 through 64. There are several attributes common to all subclasses of bioReactant, while each subclass has one or two specific attributes. Every bioReactant (1408 or 1419) has its menu (1409 or 1420), and selecting its "table" option (1410 or 1421) displays its table of attributes (1411 or 1422), in this case showing the default values for their attributes. New and not previously described attributes specific for any of the subclases include:

- **Stoichiometric-coeff** (1412) is a simple float attribute of all subclasses of bioReactant and bioProduct, which constant value represents the characteristic stoichiometric coefficient of that participant in that particular bioProcess. In other words, the set of the stoichiometric-coeff of all bioReactants and bioProducts of a particular bioProcess indicates the relationship between the number of units of each bioProduct produced per number of units of each bioReactant;

- **Alpha-coeff** (1413) is a simple float attribute which value may optionally be set by the modeler to modify by a factor the Contribution, which value is computed by the program. This is of interest for testing the effects of a global modification of such variable on the simulation of the system. This value is incorporated by default in the generic formulas provided, and if not configured by the modeler, multiplying by its default value of 1.0 has no effect on the system. Alternatively, the modeler may delete the alpha-coeff-par and provide instead a simple float value

- **Contribution** (1414) is an attribute of every subclass bioReactant which value is given by a contribution-pvar (1415), which value is dynamically computed while a simulation is running. The meaning and use of this attribute is a novel teaching of this invention, designed to meet some of the challenges posed by the domain of this invention, where both qualitative and quantitative knowledge is frequently scattered and incomplete. This attribute represents, as its name indicates, the Contribution of each bioReactant to determining the overall rate of the transformation that the bioProcess, as represented by the Velocity of its bioEngine. the contribution is a variable that can take values from 0.00 to 1.00, and represents a dimensionless scaled concentration (or other equivalent quantity variable). The scaling of the concentration is done at the bioReactant level, rather than at the bioPool level, to allow for additional flexibility. In this way, the absolute concentration can be scaled differently for each bioEngine, centering around the specific constant that characterized the molecular interactions between each specific pair of bioReactants. The type of constant may be different for different bioReactants representing the same BioPool depending on the role played by each bioEntity in a given bioEngine, such as a Km for each substrate, the Ki for an inhibitor, the Ks for ligands or complex-subunits. The default values for all types of contribution are equal to: a) 0.5 for normal steady-state physiological conditions (for that particular compartment); b) 0.0 for bioPools of inducible proteins or non-physiological molecules; or c)1 for constitutive and repressible proteins. The user can either directly override a relative value or leave the default values. A generic formula specific for each of the bioReactant classes determines how the value of each contribution class is computed.

- **Consumption-Rate** (1416) is an attribute of every subclass of bioReactant which value is given by a dependent consum-rate-pvar (1417), which value is dynamically computed while a simulation is running, and indicates the rate of consumption of units of the connected bioPool in such bioProcess. Its arguments are the bioReactant's stoichiometric-coeff and the Velocity of the bioEngine, and its value is an argument for the Output-Rate of the connected bioPool. For those classes of bioReactants that are not consumed in a reaction, the value of this attribute remains 0.0. In this invention, an enzyme.r or an inhibitor.r are preferably not consumed in the enzyme-processes. However, they may be retained, by modeling them with a time-delay-var with its specific formula. Note that the connected bioReseryoir may be configured for the quantity of the bioPool to decay dynamically, representing their degradation and diffusion components, and it also may be consumed in oth er bioProcesses;
- **Effective-binding-sites** (1418) is a simple float attribute of the subclasses of bioReactant that refers to molecules or complexes that have binding activity, such as all the subclasses of amplifier-bioreactant and leading-bioreactant, and its value is set by the modeler to indicate the number of effective binding sites per unit.
- **Kinetic-parameters** are by default simple float attributes specific for some subclasses of bioReactants, which values are optionally set by the modeler to indicate the value of the kinetic parameter characteristic for that particular instance. A set of two attributes is provided for each of such subclasses, one to hold the absolute value and the other to hold a scaled value of such kinetic parameter, which are used by the set of absolute generic simulation formulas used in conjunction with the absolute-valued variables, or by the currently default set of scaled generic simulation formulas used in conjunction with the scaled-valued variables, respectively. Alternatively, in cases where the value the **Kinetic-parameters** are by default simple float attributes specific for some subclasses of bioReactants, which values are optionally set by the modeler to indicate the value of the kinetic parameter characteristic for that particular instance. A set of two attributes is provided for each of such subclasses, one to hold the absolute value and the other to hold a scaled value of such kinetic parameter, which are used by the set of absolute generic simulation formulas used in conjunction with the absolute-valued variables, or by the currently default set of scaled generic simulation formulas used in conjunction with the scaled-valued variables, respectively. Alternatively, in cases where the value the kinetic-parameter is time-variant and dependent on other variable values, the modeler may choose for individual bioReactants to define for that attribute an instance of kinetic-parameter-var (Table 13), in which case the modeler has to define the specific simulation formula to provide the value for that particular instance, as previously described (FIG.10).
- **Equilibrium-dissociation-constant** (1423) and **Scaled-equil.dissoc.k** (1424) are simple float attributes of the subclasses of binding-bioreactant (Table 98) which value is optionally set by the modeler to indicate the absolute or a scaled value, respectively, of the Ks characteristic of that agonist or complex-subunit;
- **Catalytic-constant** and **Scaled-catalytic.k** (not shown) are simple float attributes of the subclasses of enzyme.r (Table 95) which value is optionally set by the modeler to indicate the absolute or a scaled value, respectively, of the characteristic kp of that enzyme (where kp is the equivalent of the units of activity per catalytic center, or kp = $V_{max}/[E]_t$ (t^{-1}));
- **Michaelis-constant** and **Scaled-michaelis.k** (not shown) are simple float attributes of the subclass substrate.r (Table 96) which value is optionally set by the modeler to indicate the absolute or a scaled value, respectively, of the Km characteristic of that substrate (where Km is the dynamic constant equivalent to the substrate concentration that yields half-maximal velocity, or Km = (k_{-1} + kp) / k_1, used when the bioEngine is treated under steady-state conditions, which the preferred treatment in this invention). This attribute may optionally be considered to represent the Ks (where Ks represents the intrinsic dissociation-constant, or Ks = [E][S]/[ES] = k_{-1}/k_1 (M) = 1 / Keq, may be used when the system modeled is under rapid equilibrium conditions, which is usually not required because of the dynamic nature of the simulation system of this invention), which may be used in a formula in conjunction with the catalytic-constant or scaled-catalytic.k, respectively, of the interacting enzyme.r; or
- **Inhibition-constant** and **Scaled-inhibition.k** (not shown) are simple float attributes of the subclasses of inhibitor-bioreactant (Table 99) which value is optionally set by the modeler to indicate the absolute or a scaled value, respectively, of the Ki characteristic of that inhibitor or antagonist (where the factor (1+ [I]/Ki) may be considered as an [I]-dependent statistical factor describing the distribution of enzyme between the E and EI forms).
- The **class bioProduct** (Table 60) has the subclass cell-bioProduct. As shown in FIG.15, a bioProduct (1518) is connected to an output p-connection at the bottom of a bioEngine, and represents the material contribution from a bioEngine to the bioPool to which it is connected. To be operational, the bioProduct-post connected to a bioProduct must have the same name as one of the bioPool-p-posts connected at the top of a bioPool. Every instance of bioProduct (1518) has its menu (1519), and selecting "table" (1520) displays its table of attributes

1521), which in this case shows the default values for its attributes. Not previously described attributes specific of this class include:

- "Production-Rate" (1522) is given by a produc-rate-pvar (1523), which value indicates the rate of production of the units that will be added to the connected bioPool, which is dependent on the bioProduct's Stoichiometric-coeff and the Velocity of the bioEngine, and is an argument for the Input-rate of the connected bioPool.

- **Class bioRole-post**

- In this invention, any reference to a bioRole-post means a bioPost to be located upon the subworkspace of a bioProcess which function is to connect a bioRole-Object to a bioPool through a bioPool-post. The class bioRole-post (Table 65) is a subclass of bioPost, from which it inherits all its capabilities, and has two subclasses: bioReactant-post (703 or 707), which together with a biopool-r-post (704 or 708) distantly connects a bioReactant to a bioPool; and bioProduct-post (711), which together with a biopool-p-post (712) distantly connects a bioProduct to a bioPool. The new attribute defined for these classes is:
- "ref-bioreservoir" is a pointer to the bioReservoir instance that contains the biopool-r-post or biopool-p-post to which it is distantly connected. The value is interactively set by selecting the "set-refs" menu option, and it is used by the program for interactive navigation through bioReservoirs and bioProcesses, as described below.

k) **BioModels**

- In this invention, any reference to a bioModel means an iconic knowledge structure which, depending of the subclass, encapsulates in its subworkspace a set of interrelated composite bioObjects, such as bioReservoirs, bioProcesses or other bioModels. BioModels are used to partitioned the Virtual Model into a modular hierarchy of subworkspaces of bioModels, and can be also described as subsystems or fragments of a larger network of pathways, which can be reused as modules of desired degrees of complexity to be combined in a variety of ways to build larger and diverse systems. BioModels are objects represented by icons, and the constants, parameters, and variables that model the system are distributed throughout their component building blocks, which function as distributed parallel processors. The dynamic aspect of a bioModel is determined by a system of algebraic and differential equations that provide the values for the encapsulated dependent and state variables. BioModels represent empirical biological models derived from experimental descriptive information and semi-quantitative or quantitative data. Very complex bioModels can be build by connecting component parts of an unlimited number of other bioModels by means of connection-posts. The icon of some bioModels, such as sequential time-compartments, may contain stubs to allow connection to other bioModels.
- The **class bioModel** (Table 66) is a subclass of bioView-Object and have no additional defined specific attributes, but inherit the attributes "names", "label", description", references", "warnings", and "toggle-state". It comprises various subclasses (only some examples of which are listed in Tables 66 through 73) which represent any physiological system at different levels of structural complexity, such as organ, tissue, cell-interaction, cell, or subcellular organelle, or any other intermediate compartment within those compartments, such as: submodel, bioReservoir-Bin, and On-Hold-Bin. A submodel schematically represents a portion of any size of the network of pathways of a larger structure, such as a cell-bioModel. The subworkspace (1831) of a submodel contains a set of related bioProcesses and also a bioReservoir-Bin (1831), which in turn contains a set of related bioReservoirs connected to those bioProcesses. However, the bioReactants and bioProducts of any bioProcess that is a component of a bioModel may be connected to bioPosts of bioReservoirs that may be located in the same or in other bioModels. As the submodel becomes larger, it may be broken down into smaller submodels by simply cloning a new submodel and transferring to its subworkspace the desired bioProcesses and to the subworkspace of its bioReservoir-bin the corresponding bioReservoirs. The connectivity of the pathways is transparent to movements of bioObjects from one bioModel to another, since they are directly connected among them through the structures upon the subworkspaces of bioReservoirs and bioProcesses.
- This architecture results in a hierarchy of bioModels encapsulated within other bioModels, which ultimately encapsulate bioProcesses and bioReservoirs, which in turn encapsulate bioReactants, bioEngines and bioProducts, or bioPools, respectively. Although this topology may appear complex, at run-time each of the variables depend on the variables of only those bioObjects directly connected, and therefore the system operates as a set of processors concurrently computing in parallel. The distant connections between bioProcesses and bioReservoirs, established through the bioPosts anywhere in the knowledge-base, interconnect the different bioModels into a dynamic multidimensional network of any unlimited complexity. This architecture facilitates the implementation of feedback and forward loops, interactions between components of different pathways, elements that are shared by different pathways, and connections between segments of pathways that are built as separate reusable modules. When one bioPool participates in feedback-loops in two or more pathways, this

bioPool may be able to 'switch' from one pathway to the other at a given point, as a result of continuously varying functions, such as different velocities or diffusion rates.

- The **class On-Hold-Bin** is characterized by having an activatable-subworkspace, which is usually maintained deactivated. The instances of On-Hold-Bin are used to store within a submodel (not shown) bioReservoirs and/or bioProcesses that are either substitutions for the bioReservoirs and/or bioProcesses contained in such submodel, or additional bioReservoirs and/or bioProcesses to extend the pathway represented by such submodel, which are appropriately connected, but which the user wants to exclude from a simulation or other uses. By having the subworkspace deactivated, all those structures are ignored by the inference engine and the simulator. The user may have those structures included or excluded from a simulation or other uses at will, as described under the Modeler-Mode heading. The table of attributes (2309) of each instance of an On-Hold-Bin allows to store and access additional information. All instances of On-Hold-Bin have either one of two alternative menu options: a) if the subworkspace of the On-Hold-Bin is activated, then the "deactivate-sw" is available, which upon selection by the user deactivates such subworkspace; or b) if the subworkspace of the On-Hold-Bin is deactivated, then the "activate-sw" is available, which upon selection by the user activates such subworkspace.
- Some of those subclasses of bioModels, such as the sequential time-compartments, can be independently activated and activated by the program during a simulation run. The efficiency of large-scale simulation applications can also be significantly improved using that mechanism, by having whole branches of the model that are not relevant at some point in time deactivated and activated when required, driven by events generated as a result of the simulation itself, or at given time intervals.
- The **class cell-bioModel** (2326, Table 68) is used to represent a cell as observed from the inside, that is as a container of subcellular compartments, that in turn may contain different layers of submodels until at the end of the compartment hierarchy, the bioReservoirs and bioProcesses are contained. Cell-bioModels are used for modeling prototypic cells of interest in a very detailed way through their functional structural components, which are constructed graphically through the various levels of other encapsulated bioModel that a cell-bioModel may contain in its subworkspace, but which at the end contain variables and parameters that allow to quantitatively simulate those models, or more realistically, constrained submodels within the cell, which can be selected by selecting any of the compartments at any desired level. The hierarchical encapsulated allows not only to display different levels of detail to focus only those parts of the system that are relevant to the user at any given time, but also allow to expand the model adding additional levels of detail whenever desired. Each cell-bioModel or any of its contained BioModels can be used as modules to build other bioModels with increasing degree of complexity. This modular and reusable building block structure gives the system great flexibility and transparency, while allowing to build very complex cell-entities.
- As shown in FIG.24, each cell-bioModel (2401) has a subworkspace (2402) upon which are represented the characteristic phases of the cell cycle (2421), that follow a repeated cyclic pathway, or the cell differentiate into a different stage (2428), or may go into apoptosis (2430), a terminal stage. The cell-phases (2318 through 2322) are subclasses of the **class cell-phase**, the main defined subclass of the class **time-compartment** (Table 69a), but other types of compartmentalization are also possible by means of instances of the class **time-submodel** (2303, 2423, 2325, 2426). Inatances of the **class biomodel-post** (2403, 2429), ant its associated methods (Table 70) allow to connect those cell-phases with another cells-phases upon other cell-bioModel, and to display them.
  - **G0-phase-compartment**: in cell biology the G0-phase represents resting-cells. In this invention, this compartment also represents the background processes of resting-cells, but in addition it also contains all those background bioProcesses that are not specific for any particular phase of the cell-cycle or which cross-over the boundaries of several of those phases. This compartment is frequently the starting point of a simulation and its bioPools are usually modeled by the default "normal" steady-state values of the variables and parameters of interest.
  - **G1-phase-compartment** represents a state of cells after they have been activated by external factors, and may be further compartmentalized into two or more time-compartments. For example: the **G1.1-compartment** represents activated-cells or early-G1-phase-cells which have recently entered the G1-phase of the cell cycle, as characterized by transcription of early-response genes, expression of new receptors; while the **G1.2-compartment** or late-G1-phase-cells, is characterized by transcription of late-response genes or increased secretion of specific cytokines or other signaling factors
  - **S-phase-compartment**: represents the state of cells in S-phase, characterized by DNA synthesis;
  - **G2-compartment**: represent cells in the G2-phase, characterized by double DNA content and before entering mitosis;
  - **M-compartment**: represents cells in the M-phase, during the mitotic process;

- **Di-compartment:** represents cells entering a new differentiation stage, when a different set of activated early-response and/ or late response genes induce permanent changes in the types of genes expressed. This layer is followed, depending on the combination and strength of signals provided by simulation events, by either the **G0-phase-compartment**, or both this and the **G1-phase-compartment**, of the next cell type in the differentiation pathway.
- **Ap-compartment**: apoptotic cells, when cells in the G1.1-layer do not receive the appropriate signals to enter either the G2.2-layer or the DI-layer. This is a terminal layer which only effect is to reduce the size of the total cell population. (Note that the total cell population size is increased at the transition after every M-layer, by an amount equal to the size of the population in that M-layer. The non-specifically-modeled overall cell death is represented by a decay parameter, set to fit the experimental data.
- The default time-duration of each of these temporal layers is set according to experimentally obtained knowledge of such duration, under specified "normal" conditions. A layer-rate-constant, τ_i, is constrained to values $-1 > \tau_i > 1$ and initially set to 0. When the effects of different combinations and strength of signals on the duration of any of the layers are measured, then the value of τ_i is adjusted to fit the target values of the experimental set --> $\tau = t_0(1+\tau_i)$.

- Each time-compartment (2404, 2423) encapsulates (2405, 2424) several spatial compartments that run in parallel to each other, and which represent the clearly defined subcellular compartments, called cell-Compartments (Table 67), such as the cell-membrane (2406), cytoplasm (2416), nucleus(2417), endoplasmic-reticulum(2418), Golgi apparatus(2419), endosomes(2420), mitochondria, and so on. It also contains a **timer** (Table 69b, 2427) with an activation time attribute given by an elapsed-simul0time-par, which can be used in the control of the deactivation of the subworkspace that contains it by inference methods based on such value. The connections between the time compartments belong to the **class cycle-path** (Table 38) and contain a number of parameters and variables to hold the values of the rate-constant and the progression rate, that can be used when developing quantitative models for simulation purposes. A number of other variables and parameters that describe among others: a) the size related attributes of the cells , and b) quantities related to population dynamics, such as those shown in FIG.3 (317 - 320), which are attributes of the cell-phases, are defined in Tables 3 through 11. A collection of inference blocks (Table 23) may also be used to control the activation and deactivation of the time compartments, at intervals that are predefined or dynamically computed depending on simulated variables are also defined. Table 88 lists as an example a set of simulation formulas for state variables that can be used to compute the dynamic changes in cell numbers that are accumulation in each of the cell-phase, when using these graphic structures in combination with a population dynamics approach. Furthermore, Table 78 lists a set of rules that can be alternatively used to control the progression through those cell-phases, when using the mechanistic approach. As the reader may realized, there are a number of different alternatives, encapsulated within those graphic building blocks, to model the complex systems at different levels, as discussed in the modeling section.

- FIG.24 shows the major states considered in the currently preferred embodiment of this invention, following the classical nomenclature and concepts generally accepted in the scientific community, the resting state is defined as the Go phase. Each of the values of the "cell-phase" attribute of a cell-bioModel, indicates which of those phases is currently activated, if any in addition to the G0-phase. While at Go, if the cell is activated by a growth factor, antigen or any other activating stimulus, the cell enters the activated state or G1, from which it can follow to other phases of the cell cycle or the differentiating states, depending on the signals received. G1 is bounded by the progression signal or cell birth and the initiation of DNA replication, and characterized by sequential expression of additional genes, protein synthesis and cell growth. S is bounded by initiation and completion of DNA synthesis. G2 is bounded by the completion of DNA synthesis and the initiation of mitosis, and characterized by further cell growth and/or protein secretion. M is bounded by initiation of mitosis and completion of cell division, usually represent about 2% of the total. Although the cell cycle is a continuous one, it can be compartmentalized into functional cycle intervals whenever it is deemed useful to better represent the dynamic simulation of the system. As shown in FIGs. 4, 28, and 29, there are switching points (406, 2803, and 2911) at early G1 where the cell has to decide between the differentiation and the cyclic pathways resulting in a variety of different patterns that depend on both the internal stage of the cells and other components in the system, external to the cells. A typical pattern results in: a) cells in the resting stage, when the steady-state conditions are assumed, then after activation by specific external signals, switching through the cycling phases one or more cycles, or back to resting, then after activation by specific external signals during one of those stages, switching to one of the next nodes in the tree, committing themselves to that branch of the tree in the longitudinal pathway, then once at the new node they may go back to a) for a period of time, in a resting stage

during which the new steady-state conditions are assumed, or, if activated by specific external signals they may go back to b) with the new steady-state conditions as the base-line.

- Each of the cell-Compartments comprises a subworkspace (2407) which in turn may contain various submodels (2408). Other local-submodels (2304) are also provided for other types of spatial compartmentalization. The subworkspace (2409) of a submodel may contain other submodels (not shown), and submodels may also contain bioProcesses (2410) or bioReservoirs (not shown). In the current embodiment, the bioReservoirs (2414) are preferentially hidden in a bioReservoirs-bin (2306, 2412, Table 66) that belongs to that submodel.

- As shown in FIG.25, instances of the **class cell-interaction** (Table 71, 2327, 2501) encapsulate in their subworkspace (2502) the interaction of two cells (2503 and 2504), that use the graphic extracellular-components (Table 72a) to interact with each other directly by means of surface-components (2505 and 2506); through the components that one secretes, secreted-components (2507), for which the other has specific receptors, surface-components (2508); and both interact with their environment through specific receptors (2509) for systemic-components (2510). As shown in FIG.26 , each of those components have a table of attributes (2616, 2620, and 2633), referring to an specific bioEntities (2624), bioReservoir (2626) and bioProcess (2628) within the cell-bioModel, and are connected to the internal the mechanistic pathways. Clicking on those icons displays their menus (2618) from which options, the user can select the display of the associated structures, as specified in the attributes. The different menu-options, and their associated procedures are listed in Tables 72c, d, and e). Tables 72b and f list functions that allows the modeler to automate the task of establishing the distant connections to those graphic structures. The interacting-cells (2503 and 2504, Table 71) in this interface do not contain a mechanistic model of its components, but rather refers to and displays another cell-bioModel that store those mechanistic models. In this way, this structure is used as an interface to design interactions between other cell-bioModel, and new cells and other connections to bioProcesses or bioReservoirs can be establish. The external components, such as serum or incubation media, or single or lumped toxic-byproducts, for which there are specified or lumped receptors are represented by a set of bioReservoirs, such as RPMI-Pool, serum-Pool, are connected to a bioProcess with a receptor-engine that has at least one bioProduct connected with one of the bioReservoirs within the cell. The function of the bioEngine is to compute the consumption an exhaustion of nutrients at a rate proportional to μ, the specific growth rate. The byproduct-Pool's concentration is used in turn to modify the value of μ by multiplying it by a parameter $0<m<1$.

- Cells are represented in this invention in two different ways: as the class cell-bioModel composed of cell-Compartments, as described above, and as the **class cell** used to represent a cell as observed from the outside and including any defined, mostly external, characteristic markers, as shown in FIG. 26, that can be used to assigning that cell to one type of cells and to distinguish it from cells of other types. The instances of class cell, with sets of stubs on the top and bottom of their icons, are used to classify cells according to a branching differentiation tree, as shown in FIG. 21, to define the many nuances between them that differentiate them from each other, particularly when following the almost continuous differentiation process. As an example, a Th2-lymphocyte is shown in FIG.26. An instance of cell may correspond to an instance of cell-bioModel, but it is expected that there are many more intermediary stages represented by the lighter cells, which represent the much lower level of detail most frequently available. BioReservoirs representing cell populations may refer in their Ref-bioentity attribute to a named instance of cell. One of the menu options for this class is "biomodel" which upon selection displays the subworkspace of the bioModel referred to by one of its attributes. **Ref-biomodel**, if any, and that option is only visible when such attribute has a value. The class **differentiation-pathway** (2335, 2701, Table 73) provide structures that allow the modeler to manually build differentiation trees, such as the one shown in FIG.25, which are based on expert-knowledge to represent the sequential progress from one stage of differentiation to the next. The different stages through which a cell lineage goes through is a continuous process, which can be simplified with the representation used in this system consisting of discrete stages that represent different stages of differentiation, defined by characteristic and measurable phenotype, that follow a tree like longitudinal pathways. The cells in those pathways are of the class cell (2602, 2705), and they contain further encapsulated detail, as shown in FIG.26. A cell contains information in a table of attributes (2605, and also encapsulated in its subworkspace, where the different icons are a cell-surface (2608), since this is a representation of the cells from the outside, and a set of components that allow that cell to refer a display characteristic markers and receptors that allow them to interact to the outside world, such as the subclasses of extracellular-components (Table 72a), the surface-components (2613), secreted-components (2617), and systemic-components (2612) in their environment to which they respond.

h) **Inference and Simulation Structures**

- Functions are statements which define a sequence of operations that are performed when the function's name and arguments appear as part of an active expression. The tabular-function-of-one-argument, a class defined

within the Shell, has an important use in the domain of this invention. It allows to deal with situations when the algebraic relationship between two variables is not known, but the experimental data is available, and straight-line interpolation is optional. Tabular functions can also be used in other expressions, such as the rule: if the |Product1|(|Substrate1|) decreases then inform the operator that "|Substrate-1| may be inhibitory at high concentrations".

- Relations are defined associations between two items which can be dynamically concluded and reasoned about, and in the current implementation of this invention, they represent physical and abstract relationships between objects, such as a-downstream-bioReservoir-of, the-downstream-BR-list-of. After being defined by the developer, relations are established transiently at run-time by the inference engine. The dynamic reasoning used for query, navigation, and simulation depend heavily in the use of relations, in addition to the graphical connections. Physical and abstract relationships between two objects may be represented through relations which, after being defined by the developer, may be dynamically and transiently created at run-time by the inference engine. The existence of relations can be concluded and reasoned about, and the existing relations for a particular object can be also be listed using the "describe" option. Most of the relations are created during the initialization set of procedures, while others are created at run-time by the sets of procedures used during query, navigation, and simulation. The creation or breaking of relations are treated as events that are forward chained to rules that refer to the relations in their antecedent. Relation definitions are shown in Table 75 .
- Domain-specific rules are used to conclude or to respond to the changing conditions in the world defined by the domain-specific knowledge structures. In the current embodiment, a rule has, in addition to its body that encodes the rule itself, a set of attributes that can be configured to establish its mode of operation and to define its restrictions, such as: whether it can be invoked by forward chaining and backward chaining, whether it may cause data seeking or forward chaining, focal classes and focal objects, priority, and its scan interval. Different types of rules may lack some of those capabilities. For example: a) whenever rules have no focal objects or classes, and cannot be invoked by forward or backward chaining, being only event driven. One such event may be the a relation being established or broken. Other such event may be moving the icon by the user, which is frequently used in this application to either trigger the automatic configuration of the value of some attribute, or the appearance of the icons, or the position of the icons; and c) initially rules are first and only invoked when a knowledge-base is started, or when a subworkspace is activated. Therefore, the situations to which those rules applied, and the conditions under which they can be invoked, can be very much constrained by the generic and specific sets of options and attributes of each particular instance.
- Formulas in this system can be defined as expressions that define the relationships of a variable or parameter to other variables or parameters, in a way that when one variable or parameter changes the formula predetermines in which way other variables or parameters will be changed, either by the inference engine, in the case of general formulas, or by the simulator, in the case of simulation formulas. Generic simulation and general formulas are created as statements on workspaces and provide values for a whole class or group of bioVariables or bioParameters, in contrast with the specific simulation and general formulas, which are attached to the definition of one variable and apply only to that variable. The definitions of generic simulation formula for sets of scaled-valued variables(Tables 83 and 84), mixed-type variables and absolute-valued variables(Table 86) will be discussed under the Simulation Mode heading.

2. **Modeler Mode: Creating the Virtual Models**

a) **Menus, Palettes and Libraries**

- The Modeler Mode is made available only to those users who have rights to build or modify the applications, in which case it is the default mode when the application is started. The modeler's tasks are all based on the basic paradigm of "Clone, Connect, Configure and Initialize", and consist of building and configuring graphical models of the structure of bioEntities, as well as to design and build graphical models of complex networks of cross-talking pathways, which result from the modeler's actions of cloning, connecting and configuring sets of bioProcesses, bioReservoirs, and bioEntities. To facilitate the modeler's tasks, the Modeler-Menu provides a way to organize and access the different components of the system.
- FIG.16 shows the **domain-menus** associated with this mode in this invention, which are shown here again in an overly expanded (pulled-down) way, for demonstration purposes. Selecting the "Help & Mode Menus" option (1601) allows access to the available user modes, which upon selection causes a change to that mode of operation and displays the specific domain-menus associated with that user mode, only for the window from which it is selected. Selecting Modeler Mode (1602) displays the contracted top layer of menu heads shown in the lower menu display, which additionally comprise the options "Palettes", "Structure Libraries", and "Pathway Libraries", which allow to access parts of the Virtual Models by selecting the desired options from the menu, which usually requires navigating through a hierarchy of walking menus. The following discussion

of this menu system also serves the purpose of discussing how the building-blocks are organized in Palettes and how the iconic models build by the modeler are organized and stored in Libraries. The difference between Palettes and Libraries is that Palettes contain domain-specific and generic building-blocks built by the developer and to be used by the modeler, while Libraries contain application-specific models built by the modeler and to be used by different types of users. Palettes are provided to allow the modeler to quickly create new instances from prebuilt generic bioEntities, bioReservoirs or bioProcesses, or any of their components, which comes with their corresponding default or no values, to be then configured by editing their tables of attributes. A facility is provided, called the Modeler-Palettes-Bin, with a button to automatically create new empty Palettes to be populated by modelers, to allow them to expand and complement the built-in library of building-blocks to satisfy their more specific needs.

- The **Palettes** provide a large number of characteristic types of bioObjects, grouped by subclasses and organized in special subworkspaces of named go-to-sw buttons organized in the subworkspaces of another go-to-sw named "Palettes-Bin". Palettes are accessible through the domain-menus, and because of their defined user-restrictions, selection of any bioObject upon a Palette in Modeler Mode implies clone, which automatically results in a clone of such bioObject (including its subworkspace and all structures upon it, and all the further levels of encapsulation contained in their subworkspaces) to be attached to the mouse-pointer and ready for the modeler to transfer it to the desired subworkspace. Selecting "Palettes" (1603) pulls down its options, which may point directly to the palettes of important classes of bioObjects and also model-blocks, such as: 1604 points points to 2001, 1606 points to 2101, 1608 points to 1901, and 1616 points to 2301. In the case of classes with large number of building-blocks, those may be grouped further in a number of palettes that can be accessed through walking menus, such as when selecting "Molecules" (1610) displays an additional pull-down menu headed by "Molecules" (1612), which options may now point directly to Palettes. Examples are shown in FIG.17, such as when selecting "Protein Components" (1614), the options of the walking menu generated lead. among others, to "Protein Domains" (1701), "Protein Motifs" (1706), and "Simple bioEntities" (1711).

- **Libraries** comprise among others two types:
  - **Structure Libraries**, are hierarchically organized workspaces (1623) containing models of specific complex-bioEntities (1624) that model the functional structure of molecules or molecular-complexes at various levels of complexity, and which can be accessed by selecting the options of the menu head "Structure Libraries" (1618). For example, selecting "Cytokine Library" (1620) displays the menu headed by "Cytokine Library" (1621), which options refer to libraries of different families of cytokines, and further selecting one such as "IL4.like Library" (1622) displays the library (1623) of the IL4 family, which contain several cytokines with similar structures, which prototype is IL4 (1624), which subworkspace contains several domains, motifs and groups. Those icons may be used to: a) be cloned and modify to build different but related structures; b) be placed in the subworkspace of protein-complexes, as shown in FIG.18; or c) they may be preferentially named and copied, as described below, and the dummy copies that refer to the original are used for several of the more complex structures.
  - **Pathway Libraries**, are hierarchically organized workspaces (1630) containing sets of related bioProcesses. such as those contained in the subworkspace of a submodel, and a bioReservoir-Bin with the bioReservoirs connected to those bioProcesses, where those bioProcesses and bioReservoirs are connected forming the nodes of a pathway that may be or not multidimensionally branched. The pathways included in the libraries are those that are common to several system, although they may require some modifications, and they can be cloned instead of being built from scratch. Alternatively, the "Pathway Libraries" (1628) menu may be used to hierarchically access any submodel of the Virtual Model, as an alternative to interactively navigate through all the layers of subworkspaces involved. The options of the menu head "Pathway Libraries" (1628) point directly to submodels which subworkspaces are displayed. For example, selecting the option "EGF.R.Pathway" (1629) displays the components of the initial specific steps of such pathway (1629) containing bioProcesses (1631) and a bioReservoir-Bin (1632). To conveniently handle large and complex pathways, the components can be partitioned into more submodels, by simply transferring related sets of bioProcesses and their corresponding bioReservoirs to the subworkspace of a another submodel, without affecting the connectivity of the pathways in any way. When the application grows larger and the number of submodel increase , those direct domain-menus options are replaced by other options that lead instead to walking menus offering as many hierarchical levels as necessary. Modelers can also clone submodels, in which all the bioProcesses organized in its workspace, together with the contained bioReservoir-Bin and all the bioReservoir organized in its workspace, are also cloned preserving the labels and any other attributes, but with no names, since names are unique. This means that all new bioReservoirs and bioProcesses have to be named (with the same basic name and different tags), and the connections have to be reestablished by naming the bioPosts and setting the ref-attributes, as discussed elsewhere for new sets.

b) **Creating New Composite Objects**

- There are several alternatives for building new generic types of composite bioObject. One method comprises the following steps: a) an instance of the desired class, such as bioProcess, is created by selecting from the menu of the object-definition of that class the "create-instance" option and transferred to a workspace; b) the subworkspace of such instance is created by selecting from its menu the "create-subworkspace" option; c) the desired set of instances of characteristic classes of visualObjects are created or cloned and transferred to such subworkspace; and d) the newly created instances are positioned and connected in specific ways, to conform to the designs specified in this invention. A preferred alternative way is to "clone" an existing composite bioObject that is similar to the desired one, and delete and/or add components, as desired. The additional generic building-blocks are added to palettes accessible through the domain-menus, and are used to create new instances of the same type by cloning such generic structures. Cloning an object in this invention means cloning also its subworkspace, and all objects upon its subworkspace, and any objects upon the subworkspace of each of those objects, and so on. The modeler can then further individualized each newly cloned instance by configuring the attribute tables of the superior object, or by configuring the attribute tables of the objects encapsulated within that superior object, either as attribute objects or as objects upon successive subworkspaces.
- Furthermore, it is possible to create transient objects at run time in two ways: a) by using the "create a <class-name>" and "transfer" actions; or b) by using the "create a <class-name> by cloning <instance-name>" and "transfer" actions, which is the equivalent to the clone option described above, and is the preferred way for composite objects. Transient objects can be made permanent by using in addition the "make <instance> permanent" action after the instance has been transferred to a workspace. This invention makes extensive use of dynamically creating objects at runtime by cloning, such as when creating dynamic pathways, dynamic query-panels, dynamic query-output-panels, dynamic entry-panels, dynamic graphs, dynamic lists, and so on. A set of partially prebuilt complex sets of structures, such as those required for the different types of panels, are organized and stored in the subworkspaces of objects called bins, and used as master structures to be cloned, and completed with additional context-dependent structures at runtime, as will be described under the General Mode and Simulation Mode headings below.

c) **Modeling bioEntities**

- The modular functional representations of structure that are novel teaching of this invention allow to represent the modular structural patterns so frequently used by nature. More specifically, the iconic structural representation is used to abstract the detailed structure of different molecules to highlight their similarities, and to show that even prototypes for different families present several similarities. Abstraction is obtained in part by selectively hiding some detail in the subworkspaces of other components. That representation focuses on components and the way they are arranged that relate to their function, to offer users visual models that facilitate their reasoning about structure-function relationships. The components of a molecule are used by the user to reason about its function, since the presence of given domains, motifs, or simple-bioEntities usually is associated with described function, therefore directing the scientist's attention to further focus on those functions. Furthermore, each of those iconic instances of bioEntities have their associated tables of attributes, where the modeler can include additional information, data, and references to access other sources of information about those models, which can provide to the user a wealth of knowledge in the form of an intuitive and interactive visual database. The knowledge extracted from those models can be integrated in this invention with other forms of knowledge, related to the function of those bioEntities, provided by the pathways related models. For example, the members of each family may be involved in similar mechanisms but result in very different and even opposing effects, due to the participation of their bioPools in different pathways. That knowledge integration is performed sometimes interactively by the user, and sometimes programmatically as a result of an interactive request by the user, depending on which of the many capabilities of this invention is used. Each superior bioEntity represents the structure of a biochemical physical entity in a particular state, such as different chemical modifications, different states of association with other entities, or the products of different mutations or artificial manipulation or quimeric constructions, which corresponds to the concept that different bioReservoirs hold molecules or complexes in different states. The structural and other types of information about the units that form a bioPool, as provided in the corresponding bioEntities, is supplemental and is not necessary for creation or navigation thought the pathways, or for simulation. This information is used independently of the pathways, or they may be used in combination, as in the complex queries.
- FIG.17 shows the **Palettes** with representative examples of master instances of some of the different subclasses of protein-domain (1701), protein-motif (1706), and simple-bioEntity (1711), accessed through the walking menu of "Protein Components" (1616). Some of those building-blocks are very generic or "empty" structures, such as the examples shown of the subworkspaces (1703 and 1708) of catalytic-subunit (1702) and spacer (1707).

respectively, while others may contain more or less detailed prototypic structures, in which case they are named master-structures, which may be referred to by the master-bioentity attribute of its copies, such as the example shown of the structure (1705) of a fibronectin.type.III-domain (1704). In addition, like any other bioEntity, each instance of these components have an associated menu (1805) from which options specific for each class can be selected to perform generic or specific tasks. Selecting the "table" option (1823) shows their tables of attributes, such as the examples shown for the tables (1710 and 1713) of a leucin-zipper-motif (1709) and a loss.of.function.protein.mutation (1712), respectively, providing options for the modeler to include additional information, which in this case is specific for each instance or copies. Clones of those instances can be combined in different ways to model, using the basic paradigm of "Clone, Connect, Configure and Initialize", at the desired level of detail, the structure of a bioEntity.

- In addition, to the standard Shell's "clone" capability which applies to every object, there is a novel domain-dependent capability associated with the class complex-bioEntity and all its subclasses. Selecting the "create-local" option (1824), which is restricted to Modeler Mode and when the bioEntity has a name and starts the et-create-local-proc (Table 89), allows the modeler to create local copies from any instance. The term clone means in this invention a complete reproduction of the original, with exception of the name, while the term copy means an incomplete reproduction that is usually stripped from the contents of the subworkspace of the original, but contains a pointer to the original and is used in conjunction with it. The master-bioentity attribute of the copies is automatically set to refer to the named master-structure. The dummy copies refer to the original when the "master-details" is requested, which subworkspace with the prototypic structure is displayed. That is, local copies of any complex bioEntity (including bioEntity-components), may have private specific information in its table of attributes and in its subworkspace, shown by selecting the "local-details" option, while it is can also display the subworkspace of its master bioEntity. Simple-bioEntities do not have a subworkspace and do not have master-bioentity attribute, and the concept of copy does not apply to them. This implementation is preferred because of the advantages of avoiding redundancy, including saving storage resources, and allowing modification at only one site when the details of the structure have to be modified.
- As shown in FIG.18, the subworkspace (1807) shown for a specific multi-unit receptor-complex (1804) contains the icons of its various components, which may be copies, such as those referring to structures in the Libraries (1808, 1815, and 1817), or may be originals, such as for the simple components or some specific complex components (613, 619 and 621 are originals within the original master structures) Upon clicking on each of those components, their subworkspaces or the subworkspaces of their master structures are displayed on the screen, showing their structural composition. The level of detail is optional, and components at different levels in the hierarchy of subclasses can be connected to each other. For example, clicking upon a subunit of the receptor (1808, a copy) shows its structure (1809, from the master) composed of domains, other simple-bioEntities, and other optional auxiliary structures, such as those that may contain: notes about its function (1810), sequence (1811), references (1812), or a GIF image of its three-dimensional structure (not shown). Any of those components may contain more detailed information, such as that displayed when clicking in one of those domains (1813), which structure (1814) is composed of other domains, spacers and other simple-bioEntities. Back to the main structure, clicking on a kinase (1815, copy) that form part of the complex, its structure (1816, from the master) may be represented by its two subunits, a regulatory-subunit and a catalytic-subunit, each containing or pointing to additional detail (not shown). Or the structure can be directly represented directly by it lower level components, such as when clicking on other associated kinase (1817), which structure (1818) is composed directly by several domains and other simple-bioEntities. Any of those molecular components may contain in their subworkspace any type of other relevant information, when more detailed structure is not available or not desired. For example, clicking on one of the domains (1819) shows (1820) differences in the sequence of a functionally important homology region of that domain for different members of that family of molecules (which allows to represent all members of the family by just one structure, by describing additionally the differences between the different members), while clicking on another domain (1821) shows more information (1822) about conserved subdomains within that family. In general, simple-bioEntities may be linked to a motif, a domain or even to a subunit or a molecule. The structure may focus on only those parts of the structure that have functional relevance. If intermediary stretches within the structure are unknown or that detail is not desired, they may be represented by domains called spacers.

d) Modeling Pathways

- The **bioReservoirs Palette** contains prebuilt default instances of
- **bioReservoir-Bin**, used to store a set of related bioReservoirs upon its subworkspace and prebuilt with the standard set of navigation buttons used on most subworkspaces, a go-to-sup-obj-button and a hide-sup-ws-button, and a bioObject-title, which is standard for the subworkspaces of all bioModels as well as of all complex-bioEntities.

- **On-Hold-Bin**. used to store upon its subworkspace bioReservoirs and/or bioProcesses connected into pathways that the modeler may want to offer as alternatives, and which the user may include or exclude from a simulation or other uses at will, by using any of three alternatives: a) desired individual structures can be moved out of the On-Hold-Bin and transferred to the subworkspace of such submodel, or its bioReservoir-Bin if bioReservoirs; b) if all structures are desired for inclusion, the subworkspace of the On-Hold-Bin can be activated and deactivated at will; or c) two On-Hold-Bins can be used to transfer those structures between them at will, one to be permanently deactivated and the other be activated and to hold only those structures desired for inclusion. If any of those structures are substitutions for other bioReservoirs and/or bioProcesses contained in such submodel, then those structures they are replacing should be removed to a deactivated On-Hold-Bin. This subworkspace is prebuilt with an additional hide-and-deact-sup-ws-button, which upon clicking on it when the subworkspace is activated, hides and deactivates such subworkspace
- a set of different **prototypes of bioReservoirs**, representing all the subclasses, which differ in the instance of the bioPool they contain upon their subworkspace, which is in turn of one of the corresponding subclasses of bioPool, and in the color of the type-color region of their icon. Each of those prototypic bioReservoirs comes with a minimal set of basic structures, and model-boxes and additional bioPosts are added interactively when needed. By cloning a bioReservoir of the appropriate class from its Palette, transferring it to the subworkspace of a bioReservoir-Bin and clicking on its icon to select various tasks, it can then be connected to any of the bioRole-Objects of the bioProcess by simply giving the same name to a matching pair of bioPosts, one from each structure, always keeping in mind the required restrictions.

- Another alternative for adding bioReservoirs to the Virtual Model is to clone existing bioReservoirs with similar characteristics, by selecting the "clean-clone" option from their menu, and configure the new name and desired attributes. The "clean-clone" option starts the BR-clean-clone-proc, which creates a clone of such bioReservoir and sets the values of all the attributes of all structures involved, including the bioReservoir and its subworkspace and all structures upon it, to their default values. This capability is useful to create clean bioReservoirs of desired types that are available on the working submodel, without the need to open the Palette . The modeler can add, modify or remove connections to bioProcesses, by creating and naming the corresponding bioPosts. That established distant connectivity is independent of the location of bioProcesses and bioReservoirs throughout the Virtual Model or the workspace hierarchy, and no additional changes are required when moving those structures from one location to another. Various menus options and associated methods are offered to automate the task of establishing those connections, which will be described after the discussion of the creation of bioProcesses

- The modeler can also model inputs or outputs to a bioPool as mathematical models using model-blocks held in **model-boxes**, as previously discussed (FIGs. 8 and 12). A set of menu options, restricted to appear only when in Modeler Mode, are defined for the class bioPool to automate the task of adding a scaled-input-model-box, input-model, or output-model-box, respectively, when desired. The options labeled "add-scaled-input", "add-input", and "add-output", start their associated methods, create-scaled-input-model-box-proc (Table 90), create-input-model-box-proc, and create-output-model-box-proc, respectively, which result in the cloning of the corresponding prebuilt master model-boxes, and then transferring the new instance to its specified position in relation to the bioPool. The modeler has only to extend its stub to join it with the color-matched stub of the bioPool, which is the closest to it. Each of those named master structures is simply a generic model-box, each of the subclass referred to by its name, with a color-coded subworkspace containing, in addition to instructions for the modeler on how to proceed, a connection-post which attribute"superior-connection" has been set to refer to the only connection stub of the superior model-box. With this construction, any model-block connected to such connection-post upon the subworkspace of a model-box is connected also with any object to which the model-box is connected through such superior connection. This implementation results in the generic ability for the modeler to establish a connection between an encapsulated model-block and a bioPool, by cloning the desired model-block from the Palette of generic model-blocks and connecting to the unique connection-post upon the subworkspace of a model-box connected to that bioPool. As described above, the generic formulas that provide the value for the Accumulation of each bioPool already integrates the output of any such connected model-block as either an absolute input, a scaled input or an output, depending on the class (each connected at its predefined specific port) of the model-box that contains the model-block.

- As shown in FIG.19, the **Model-blocks Palette** (1901) is provided to the modeler, each being a default instance of each of the subclasses of model-block. Model-blocks (Table 22) and the different generic simulation formulas (Table 67) to compute their value of their output-1 attribute have been previously defined. Some examples of the attributes for some subclasses with their default values are shown through the tables of attributes of: constant.f (1902), exp.c.pdf (1903), exp.m.pdf (1904), unif.c.pdf (1905), sigmoid .f (1906), and generic.model.block.f (1907). The latter is a very generic block that allows the modeler to create instances and configure as many attributes as required by defining their slots with any desired type of value, including instances of model-block-var, in which

case their specific formula would also have to be defined. The output-1 of this block is also a variable for which the modeler has to write its formula. to incorporate as arguments the newly configured attributes

- As shown in FIG.20, the **bioProcesses Palette** provides the most representative types of generic bioProcesses, prebuilt and ready to be cloned and further configured. These preassembled and stored structures allow the repeated use of bioProcesses as building blocks of more complex pathways. Like in chemical systems, some bioProcesses are relatively simple and some may become more complex, but all are built with the basic building blocks (2504 through 2511). The modeler can, for example, clone a R2.L1, which subworkspace (2008) comprises one receptor-bioengine connected with two receptor.r (which may each be connected to different bioPools, if the active receptor is an homodimer of two equal molecules, or to the same bioPool, if an homodimer of two equal molecules), and one ligand.r at the top, and one bioProduct at the bottom. After transferring the clone to the desired workspace, the bioProcess can now be configured, by clicking on its icon and selecting its table of attributes. Selecting "details" from its menu displays its subworkspace and provides access to all its components, which can now be configured by accessing the tables attributes of the bioEngine, each bioReactant and each bioProduct, and each bioReactant-post and bioproduct-post. BioProcesses have activatable-subworkspaces, which means that any object upon such subworkspaces are not recognized by the inference engine or the simulator unless the subworkspace is activated. When new bioProcesses are cloned, their subworkspace are deactivated by default, and have to be activated selecting the "activate-sw" menu option available only when the subworkspace is deactivated.
- An alternative to the Palette is to clone another existing bioProcess with similar characteristics and labels directly from a submodel, and then transferring them to the desired submodel and the configuring its components. Two menu options are provided to facilitate that task labeled "labeled-clone" and "clean-clone", which start either BP-labeled-clone-proc or BP-clean-clone-proc, respectively. The first procedure creates a clone of the bioProcess, and then sets the values of all the attributes except the label of all structures involved, including those of the bioProcess and of all the structures upon its subworkspace, to their default values. The second procedure calls the first one, and also sets the values of the labels attribute of all structures involved to their default values.
- The **bioEngines Palette** and the **bioReactants Palette** shown in FIG.21 are provided for the purpose of allowing the modeler to: a) modify individual instances of bioProcesses cloned from the Palette, to accommodate particular infrequent needs; and b) to design new prototypic bioProcesses, by starting from a clone of an existing bioProcess and adding or removing bioReactants or bioProducts, or by exchanging the bioEngine for another of compatible type. The modeler may want to modify the Velocity attribute of any bioEngine, and to provide specific formulas to represent their favorite equations to best describe the process being modeled, which have to refer to the specific set of bioReactants connected to that bioEngine. Such newly designed bioProcesses may be stored as building-blocks in the appropriate Palette within the Modeler Palette Bin, as described.
- In addition, the modeler may use a tabular-function (Table 74) as a look-up-table, from which the program is able to interpolate, to model a process by entering experimental pairs of values of the independent and the dependent variables in such table, such as a dose response. To facilitate that task, the menu option "create-sw" is defined for the class bioEngine that starts the create-sw-for-bioengine-proc, which creates a subworkspace for the bioEngine by cloning the subworkspace of the bioengine-sw-master, which contains a tabular-function-of-1-argument and instructions for the modeler about how to use such a tabular-function.
- As shown in FIG.22, several steps are required **to establish the connection** between a bioReactant or a bioProduct and a bioPool. To facilitate this process, and more importantly, to reduce the sources of error while entering those values by the modeler, various methods are offered to automate the tasks of: creating and naming, or removing the names, of bioPool-posts, removing the names and the references of of bioRole-posts, and to set the appropriate values for the ref-bioprocess of bioPool-posts and ref-bioreservoir of bioRole-posts, as following:
  - The first step is to select the bioProcess (2201) and bioReservoirs (2205, 2207, and 2209) to be connected, if already in existence, otherwise they are created by cloning from the Palettes and named. BioProcesses and bioReservoirs have to be named before their bioPosts can be named. By selecting the "details" option (2203) from the menu (2202) of the bioProcess, and such of the bioReservoirs (not shown), their subworkspaces (2204, 2206, 2208, and 2210, respectively) of each of them are displayed.
  - the menu option "add-top-post" (Table 92) defined for the class bioPool starts the create-biopool-p-post-proc (Table 92), which first scans for the first free stub of those defined at an even-numbered port (p02 to p12) and a) if one is found, it creates an instance of biopool-p-post, and aligns it opposite to such stub within the first layer of such biopool-p-post, b) if none is found, it then scans for the first free stub of those defined at an uneven-numbered port (p01 to p11) and, if one is found, it creates an instance of biopool-p-post and aligns it

opposite to such stub within a second layer (more distant from the bioPool) of such bioPool-posts, and if none is found, then: i) it moves the existing biopool-p-post connected at the first port (p01) to a more distant position, to start the next layer, ii) it creates an instance of biopool-p-post and transfers it to the next free predefined position within that layer; and c) it sends a message to the modeler with instructions for making the specific connection. Then it calls the pool-post-name-it-proc (Table 93) and returns. The "add-bottom-post" functions in a similar way, but it adds instead biopool-r-posts at the bottom of the bioPool.

- The next step is to connect each of the bioRole-posts (2212, 2218, and 2225) to a complementary and not yet connected bioPool-post of the appropriate bioReservoir. Here the first pair (2211 and 2212) has been already connected, which is not apparent to the user. In most cases, this process will first require to create a new bioPool-post of the appropriate subclass, by selecting from the menu (2214) of the bioPool (2213) either the "add-top-post" option (2215) or like here, the "add-bottom-post" option (2216), which results in the automatic creation of an appropriately named bioPool-post, and its transfer to the appropriate position (2217), which is close to the next available free stub with the same cross-section pattern on the bioPool. The modeler has now to establish the connection to the bioPool by extending its stub and joining it to the closest free stub on the bioPool, and then establish the distant connection to the bioReactant-post (2218). Here is shown how such distant connection is established between and already named and connected bioPool-p-post (2219) and the bioProduct-post (2225). Clicking on the bioPool-p-post (2219) in Modeler Mode displays its menu (2220), and selecting the "table" option (2221) shows its table of attributes (2222). Note that the tables shown here for a bioRole-post (2222) and a bioPool-post (2228) are shown with all the values already set, as they would appear for fully functional bioPosts. However, at this point in the connection process, the values shown for the "ref-bioprocess" (2235) would be the default value none, while the values of the "names" (2223) and the "label" attributes (2224), will be already set. The value of the "label" attribute, is set to be the same as its name in the case of all bioPosts , and is used at run-time when using the alternative simulations procedures, as discussed under the Simulation Mode heading.
- The next step is to give the bioRole-post (2225) the same name as the name of the bioPool-post (2219) to which is to be connected. Here, this is accomplished by selecting either the "name" menu option (2227) or the "names" attribute value slot (2232) of the bioRole-post (2225) and clicking on the name-display (2228) of the bioPool-post (2219). Note also that any time the name of any object is edited, its name-display (2228) appears in the proximity of that object, and that name-display has text-insertion capabilities, in a way that whenever the text-editor is opened (which happens among other occasions when clicking on the "name" option of any object or on the value slots of the tables of attributes), clicking on such display automatically inserts its text into the slot or at the position of the cursor. In the current implementation of this invention, most name-displays are hidden by the developer or modeler after they are used, since the labels are used instead to provide a more flexible and user friendly method of recognition for those objects when one is desired. In the case under discussion, all the names of bioRole-posts are hidden, while only the name-display of the last bioPool-post is left as a means of also providing a reference to the name of the superior bioReservoir.
- The previous step completes the basic connection process that allows the inference engine and the simulator to reason about those connections. The additional step that follows adds information to those connected bioPosts that is used by the program to quickly access the components on the subworkspaces of those connected bioReservoirs or bioProcess, even when the subworkspaces of those connected bioReservoirs or bioProcess are deactivated, which may be the case when in Simulation Mode, and which would cause the system to not recognize those connections, since the system would not recognize the connected bioPosts upon such subworkspaces. A method is provided to facilitate and reduce the sources of error in setting such additional information. Once the bioRole-post (2225) and the bioPool-post (2219) are equally named, the "ref-bioreservoir" attribute (2234) of the bioRole-post and the "ref-bioprocess" attribute (2235) of the bioPool-post are automatically set to refer to each other's superior bioObjects, as their names indicate, by just selecting the "set refs" menu option (2229) of the bioRole-post, which is only available when it has a name. The "ref-bioreservoir" and "ref-bioprocess" attributes are in turn used by the program when the user selects the "show-br" option (2241) or the "show-bp" option (2242), which are available in the menus (2220 and 2229) of bioRole-posts and bioPool-posts, respectively, only when those attributes have a value different than none.
- Alternatively, connections can be changed and previously connected bioPosts can be recycled by using consecutively the following set of menu options: "unname-it" (2219) of the bioPool-post which only appears if it has a name, "unname-it" (2229) of the bioRole-post which only appears if it has a name, and "name-it" (2223) of the bioPool-post which only appears if it has no name, which tasks are obvious and described below, followed by clicking the name of the bioPool-post into the name slot of the bioRole-post and selecting the "set-refs" (2228) of the bioRole-post, as described above for new connections.

- A more detailed description of those methods, with references to their definitions follow:

- the menu option "name-it" (Table 93) associated with the class bioPool-post is only available when the bioPool-post has no name, and upon its selection by the modeler the pool-post-name-it-proc is started. This procedure gives the "names" attribute of the bioPool-post a name composed of the base name of the superior bioReservoir (removing the ending BR tag) followed by the default value of the "label", which is a reference to the class and relative position of that bioPool-post, such as PP4 referring to bioPool-p-post connected at port p04, and also colors the face-color region of its icon.

- the menu option "unname-it" (Table 94) associated with the class bioPool-post defines an option which is only available in the menu when the bioPool-post has a name. Upon its selection by the modeler it starts the poolpost-remove-name-proc, which changes the values of the names and the ref-bioprocess attributes of the bioPool-post to their default value none, removes the base-name from the value of the "label" restoring it to its default value, and resets the face-color of its icon to gray.

- the menu option "set-refs" (Table 96) is associated with the class bioRole-post. It defines an option which is only available in the menu when the bioRole-post has a name, and upon its selection by the modeler it starts the rolepost-set-refs-proc, which changes the values of the "ref-bioreservoir" of the bioRole-post and the "ref-bioprocess" of its connected bioPool-post to equal the names of the bioReservoir superior to such bioPool-post and the bioProcess superior to the bioRole-post, respectively.

- the menu option "unname-it" (Table 95) associated with the class bioRole-post defines an option which is only available in the menu when the bioRole-post has a name. Upon its selection by the modeler it starts the rolepost-remove-name-proc, which changes the values of the "names", "label", and "ref-bioreservoir" attributes of the bioRole-post, and the "ref-bioprocess" attribute of the bioPool-post, to none.

- The metabolic and regulatory pathways of cells and organisms are composed of a large number of processes. To simplify the overall structure of the system, or when the intermediary detail is unknown, it is possible in the currently preferred embodiment of this invention to consider parts of such pathways as black-box-bioProcesses with inputs and outputs connected to other modeled pathways or to other black-box-bioProcesses, which are not modeled in detail. If and when it is desired to model disturbances at specific points in the pathway represented by a black-box-bioProcess, it can be split into the pathway before and the pathway after the regulated location, which is represented by a normal bioProcess, with the two new pathway segments now represented by two new black-box-bioProcesses, and the segment where the disturbance occurs to be defined and modeled.

- In addition, it may be desirable to model a pathway as a single lumped bioProcess. Such may be the case for an enzymatic pathway for which: a) the details may be unknown or not desired, b) only the inputs and the outputs are relevant, and c) sufficient quantitative or qualitative information is available about the rate limiting enzymes and the inputs. Because of metabolic and energy/mass conservation constraints, only a few of their net conversion rates are independent variables, and the others are mutually dependent, various of those reactions can be combined and the pseudo-steady-state approximation of the combined process can be used instead. For these cases, the currently preferred embodiment of this invention provides enzyme-bioProcess subclasses with two or more enzyme-bioReactants, with their Velocity depending only the Concentrations of those enzymes, together with the substrates that are inputs to the pathway and the products are outputs of the pathway, ignoring the intermediaries and other participating enzymes. The following are examples of overall rates or synthesis resulting in the production of a cytokine, IL2, as a cellular response to its binding of another extracellularly provided cytokine, IFN-α, that could be represented by a set of bioReservoirs and bioProcesses, to represent each of the terms of the equation:

  - $rs(TF\text{-}IFN\text{-}\alpha) = k_1 * [IFN\text{-}\alpha]$ and
  - $rs(mRNA\text{-}IL2) = k_2 * [TF\text{-}IFN\text{-}\alpha]$; or $rs(mRNA\text{-}IL2) = 1 + k_a [IFN\text{-}\alpha] / 1 + k_a [IFN\text{-}\alpha] + k_s$;
  - $rs(IL2) = k_3 * [mRNAIL2]$;

e) Biochemical Modeling Techniques

- To be connected to each other and integrated into pathways, bioReservoirs and bioProcesses must have unique names.

- For bioReservoirs that represent the input into the system of a synthetic agonist or antagonist: a) the Concentration has: a) a normal default initial value of 0.0, b) has no modeled inputs from bioProcesses, and c) it may be modeled by a model-block. However, quantities of that compound may be transferred to another bioReservoir in a different compartment, and that latter bioReservoir is a regular one.

- It may not be desired in most occasions that the synthesis of macromolecules, such as DNA, RNA, and proteins be visually modeled by bioProcesses, because of its complexity and the large number of steps or components involved. In these situations, the synthesis component may also be modeled by adding a model-block to one input-box connected to the bioPool, which is represented by the additional term already integrated in the Accumulation equation. The output of this model-block may define a global synthesis-rate (rs(t)) that may be made dependent on the values of any other variable of the system multiplied by a synthesis-rate-constant (k_{rs}) parameter, a separate attribute of the model-block. The k_{rs} may also be dynamically increased or decreased (by multiplying by appropriate factors) as defined in rules or procedures, which are fired or called as a result of the activation or deactivation of other bioProcesses, which directly or indirectly regulate the processes of DNA-transcription into mRNA, mRNA-splicing, mRNA-translation into protein precursors, or the processing of the precursors into the active-form of the proteins (all or part of which may or may not be visually modeled).
- When degradation or diffusion are important components of the model, they can be separately modeled by: a) using additional bioProcesses to model those processes; or b) adding a model-block to the output-box connected to the bioPool; or c) using additionally defined degradation-rate or diffusion-rate parameters, respectively, and modifying the formula accordingly by adding the corresponding terms. The first two approaches are recommended because they are more flexible and modular.
- Processes in which the binding step is the limiting factor (the catalytic or other actions are fast so that the ES complex is very short-lived), are usually modeled by a single bioProcess. If the complex stays around exposed to other interactions, two bioProcesses may be used: a) complex-formation and b) catalytic or other action, each with its respective constants. Inhibitors and antagonists should as frequently as possible be modeled by separate bioProcesses.
- In the system of this invention, the intermediary complexes that are formed in enzymatic reactions, and their concentrations such as [ES1] or [ES1S2], are preferably not explicitly represented, unless they are the specific targets of regulation or translocation to different compartments. Otherwise they are implicitly represented within the model equation of the bioEngine.
- Reaction systems with two or more substrates of the ordered or ping-pong types, may be represented by either one bioProcess with two substrate.r, in conjunction with the appropriate Velocity equation, or by two separated bioProcess in series. Preferable, these types of reactions are modeled by two bioProcesses in which the participants are represented as following: a) the enzyme and the first substrate are the enzyme.r and substrate1.R of the first bioProcess, b) the bioPool of the enzyme-substrate1 complex is represented by both a bioProduct of the first bioProcess and by the enzyme.r for the second bioProcess, c) the second substrate is the substrate.r of the second bioProcess, and d) one bioProduct of the second bioProcess may be represent the bioPool of the enzyme. To be more accurate, in the ping-pong case the bioProcess cannot be said to be first or second, since the enzyme oscillates between the "free" and the "complexed" states.
- In reactions with cooperativity, such as allosteric reactions of an enzyme with n equivalent substrate binding sites, can be either modeled by: a) one bioProcess with a 1E.nS.1I.HMME bioEngine with each of the n substrate.r representing the same bioPool, each with a stoichiometric coefficient equal to 1, but each with a different effective Km corresponding to each additionally bound molecule, and a Velocity formula representing a Henry-Michaelis-Menten equation, as shown in ----, or b) one bioProcess with a 1E.1S.1P.0I.HE bioEngine with one substrate.r with stoichiometric coefficient equal to n, and the Velocity formula representing a Hill equation, as shown in ----. In that case, the overall affinity constant ($K = \varphi^*K'^{(n)}$, where $\varphi = \alpha^*\beta^*\delta^*\ldots\xi$) is equivalent to the product of the K_i' of the first bound site times the series of K_i' times the coefficients by which each molecule of ligand bound modifies the affinity of the other sites.
- Allosteric reactions of an enzyme with binding sites for two or more different ligands, either substrates, inhibitors or modifiers, can be either modeled by: a) one bioProcess in which each ligand is represented in the formula for appropriate class of variable providing the Velocity, or b) two or more bioProcesses, in series or parallel or both, when the mechanism of the reaction is known and information is available. The latter case is preferred where separate bioProcess, one for each ligand, compete with each other for the common component(s), with an strength dependent on their respective affinities and Concentrations. In addition to the cases above, it is in general preferable to model several complex-form-process in the following cases, as described in more detail below:
  - all types of inhibitions in which an irreversible complex is formed, such as E + I -> EI, or E + S + I -> SEI,
  - the effects of antagonists, such as R + Ant -> R-Ant, and
  - the feedback activation or inhibition by products, such as E + X + P -> P + Y + Z, where P is the product of either this reaction or a downstream reaction.

- Different types of inhibitors may be modeled in a variety of ways to show different behaviors, usually involving a complex-form-process in which the resulting complex would have different characteristics:
  - A competitive inhibitor and the substrate are mutually exclusive, which means that EI does not bind S, but it may result in E + S + I <-> ES + EI-> P . This type of reversible competition is best modeled as one enzymeEngine comprising enzyme.r, substrate.r and inhibitor.r, and a bioProduct, with its matching standard kinetic equation. The intermediary complexes may be modeled if desired by editing the value of the time-lag, to account for the time the E, S and I spend in the complex forms ES and EI, not explicitly modeled otherwise. The same approach can be applied to any other class or type of bioProcess, when appropriate.
  - An irreversible inhibition is equivalent to enzyme removal from the system, with the Vmax reduced, and it is therefore best represented by two parallel bioProcesses, of different classes: a) an enzymeProcess for E + S -> P, and b) a complex-form-process for E + I -> EI. The production rate of P is influenced by the Concentration of S, directly, and I, indirectly, while the production rate of EI is indirectly influenced by the Concentration of S. It may also be represented by one enzyme-process containing enzyme.r, substrate.r and inhibitor.r, and two bioProducts (E + S + I -> P + EI), with its matching standard kinetic equation.
  - An uncompetitive inhibitor may bind reversibly to the ES complex yielding inactive ESI complex, but it does not bind to enzyme alone. This model needs some modifications of the regular bioProcesses, because it requires access to an intermediary complex that is not explicitly modeled in the system of this invention in a standard enzyme-process (E + S + I -> P). This type of reaction may be represented by three bioProcesses, first in series and then in parallel: a) complex-form-process: E + S -> ES ; b) enzymeProcess: ES -> E + P; and c) complex-form-process: ES + I -> ESI. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r and inhibitor.r, and one bioProduct (E + S + I -> P), with its matching standard kinetic equation.
  - A noncompetitive inhibitor and the substrate do not affect each other's binding to the enzyme, however the trimeric complex SEI (ES + I -> SEI or EI + S -> SEI) may be inactive. The main effect is a sequestering of either enzyme or both enzyme and substrate, and the Vmax is reduced. This type of reversible competition is best modeled by three different bioProcesses: an enzyme-process E + S -> P (a), and two complex-form-process: E + I -> EI (b), and E + S + I-> SEI (c), where (a) and c) have the same Ks and (b) and (c) have the same Ki. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r and inhibitor.r, and three bioProducts (E + S + I -> P + EI + SEI), with its matching standard kinetic equation.
  - A mixed-type inhibitor I is such that reversible binding of either I or S to E changes the Ks or Ki of the other by a factor α, yielding ES, EI and inactive ESI complexes. It may be best represented by three different bioProcesses: an enzyme-process E + S -> P (a), and two complex-form-process: E + I -> EI (b), and E + S + I -> SEI (c), where (a) and (c) have different Ks and (b) and (c) have different Ki. The Ks' of (c) may equal the Ks of (a) multiplied by α, or similarly, the Ki' of c = α*Ki, where α is a cooperativity-coefficient that may be introduced as a factor in the formulas of the respective Ks or Ki.
- Different types of feedback regulation by downstream products is represented by different sets of bioProcesses. As examples are given the following cases of feedback inhibition by more than one product:
  - Cooperative or synergistic inhibition is when mixtures of P_1 and P_2 at low concentrations are more inhibitory than the sum of inhibitions by each at those concentrations, and at saturating levels of either one of the products P_1 and P_2, the velocity of the reaction may be driven to zero. It means that both P_1 and P_2 can combine with E simultaneously to form EP_1P_2 complexes. This system may be represented by 4 bioProcesses: an enzyme-process: E + S -> P (a), and three complex-form-processes: E + I_1 -> EI_1 (b), E + I_2 -> EI_2 (c) and E + I_1 + I_2 -> EI_1I_2 (d), , where $I_1 = P_1$, and $I_2 = P_2$, and where (d) have different Ki_1 and Ki_2 than (b) and (c) (the $K'i_1$ of (d) = α*Ki_1, and the $K'i_2$ of (d) = β*Ki_2, where α and β are cooperativity-coefficients).
  - Cumulative or partial inhibition is when each end product is a partial inhibitor, and a saturating level of one alone cannot drive the velocity to zero. Each of I_1 or I_2 alone, either increases the Ks of S (partial competitive inhibition) or decrease kd of E (partial non-competitive inhibition) or both (partial mixed-type inhibition). True cumulative inhibition results in the second case, where the total inhibition factor is $i_T = i_1 + i_2 *(1-i_1)$. This type of reversible competition is best modeled by three different bioProcesses: an enzyme-process: E + S -> P (a), and two complex-form-processes: E + I_1 + I_2 -> EI_1I_2 (b), and E + S + I_1 + I_2 -> ESI_1I_2 (c), where (a) and c) have the same Ks and (b) and (c) have the same Ki_1 and Ki_2. It may also be represented by one enzyme-process comprising enzyme.r, substrate.r, inhibitor1.r and inhibitor2.r, and three bioProducts (E + S + I_1 + I_2 -> P + EI_1I_2 + ESI_1I_2), with its matching standard kinetic equation.
  - Concerted or multivalent inhibition is when both end products have to be present to be inhibitory, and either alone has no effect on E. It means that either EI_1I_2 does not bind S or ESI_1I_2 is inactive. It may be represented in

two alternative ways, either by: 1) two bioProcesses: one enzyme-process: E + S -> P (a) and one complex-form-process E + I1 + I2 -> EI1I2 (b); or 2) one enzyme-process: E + S + I1 + I2 -> ESI1I2 (c). In both (b) and (c), the matching standard kinetic equations include a term to represent the concerted effect of I1 and I2, that is proportional to the Concentration of I1 multiplied by the Concentration of I2.

- Additive inhibition implies the presence of two distinct enzymes or catalytic sites, each sensitive to only one of the feedback inhibitors. This type of reversible competition is best modeled by three enzyme-processes: a) E + S1 + S2 -> P; b) E + S1 + S2+I1 -> P; and c) E + S1 + S2 + I2 -> P. It may also be represented by one enzyme-process comprising enzyme.r, substrate1.r, substrate2.r, inhibitor1.r and inhibitor2.r, and one bioProduct (E + S1 + S2 + I1 + I2 -> P + EI1I2 + ESI1I2), with its matching standard kinetic equation.
- Sequential inhibition is when downstream of E1 and S1, one product P1 inhibits enzyme E2 and another product P2 inhibits enzyme E3 of another pathway of utilization of S1. When both P1 and P2 ==are at high levels, S1 accumulates, and S1 may also inhibits E1. This type of feedback inhibition is best represented visually by a combination of the appropriate bioProcesses, as described above, in parallel and in sequence as required.

- Lumped Enzymatic Pathways: In occasions, it may be desirable to model a pathway, such as an enzymatic pathway, for which: a) the details may be unknown or not desired, b) only the inputs and the outputs are relevant, and c) sufficient quantitative or qualitative information is available about the rate limiting enzymes and the inputs. Because of metabolic and energy/mass conservation constraints, only a few of their net conversion rates are independent variables, and the others are mutually dependent, various of those reactions can be combined and the pseudo-steady-state approximation of the combined process can be used instead. For these cases, the currently preferred embodiment of this invention provides enzyme-bioEngines classes with templates and stubs for two or more enzymes, and which formulas consider only those enzymes, together with the substrates that are inputs to the pathway and the products that are outputs of the pathway, ignoring the intermediaries and other participating enzymes.
- Black-box-bioProcesses: The metabolic pathways of cells and organisms are composed of a large number of chemical reactions and compounds, that, assuming normal physiological conditions, maintain the steady-state of the system. To simplify the overall structure of the system, in the currently preferred embodiment of this invention many of such metabolic pathways are considered as a black box with inputs and outputs connected to other modeled pathways or two other black boxes, and are generally not modeled. The black boxes are considered, until better defined, as a collection of constitutive bioEntities interacting under steady-state conditions. If and when it is desired to model disturbances at specific points in the pathway, the black-box can be split into the pathway before and the pathway after (both of which remain like black boxes), and the segment where the disturbance occurs to be defined and modeled.

f) B Handling Challenging Aspects of Modeling Complex Systems' Behavior

- **Modeling Internal Mechanisms of Cells**

- The transfer of molecules from compartment i to compartment j does not happen at once, but rather follows a Gaussian distribution, which may be modeled using the corresponding model-block to from which the translocation-rate would be dependent. The bioEngines have also a "time-lag" attribute, defined as the time interval between the availability of the bioReactants to the bioProcess and the the availability of the bioProducts to the next bioPool, which can be used for processes where translocations of molecules or cells are involved. The analytic equations for the distribution of bioEntities in consecutive pools, are a function of the initial concentration of the first bioPool and the corresponding translocation-rate constant (k) and time-delays (d), respectively, of the successive bioEngines. For example, to model the series of translocations of proteins through the several compartments in which they are processed after their synthesis, with the system of this invention, the modeler would clone from the Palettes set of bioReservoirs and bioProcesses of the appropriate types o represent the different terms of the following equations, which are then transferred to the appropriate compartments, connected and configured, and desired values are given to the model parameters. Translocation rates between compartments could be expressed as translocation half-times, defined as : $t_{1/2,nu} = \ln(2/K_{nu})$, where K_{nu} is the rate constant for processing in the nucleus or the rate constant for transport from the nucleus to the ER, or both lumped together. For steady-state balanced growth, where $\mu=0$ and rp(t) is the rate of transcription in the nucleus (molecules/hour/cell): $[RNA]_{nu,ss}(t) = rp(t)/K_{nu}$. The additional equations that follow are similar to those proposed for cell cycle-dependent protein accumulation by Bibila & Flickinger, (1991). Biotechnology and Bioengineering 38: 767-780, and have to be adapted for use in the simulation formulas of the system of this invention which takes a dynamic approach:
- $[X_1] = \bar{s}X_1 * [X_1]_0 * e^{-k1*t}$ (this is ER)

- $[X_2] = \S X_2/\S X_1 * [X_1]_0 * (e^{-k1*t'} - e^{-k2*t'}) * k_1 / (k_2-k_1)$, where $t'=t-d_1$ (this is Golgi)
- $[X_3] = \S X_2/\S X_1 * [X_1]_0 * (-(e^{-k1*t''}-1)/k_1 + (e^{-k2*t'}-1)/k_1)/(k_2-k_1)$, where $t''=t-d_1-d_2$ (this is extracell. comp. for secreted proteins)
- On many occasions, the newly generated molecules are secreted, while on other occasions, the newly generated molecules are retained in the cell, either to maintain housekeeping functions, which are usually included in the steady-state and no implicitly modeled, or they result in new cellular functions, in which case, its mechanisms are graphically modeled. If this response results only in proliferation, the inference engine may monitor the simulation of the cell mechanisms, and based on the values of certain variable or sets of variable, may cyclically activate the next phase-compartment in the cycle. However, if the response results in cellular differentiation, the inference engine activates a different cell-bioModel, if such exits, that represents the new stage of differentiation, or will send a message to the user otherwise.
- The different bioProcesses in a cellular bioModel may be activated for different periods of time, if placed within sequential time compartments, but they will be overlapping with many other bioProcesses contained in the G0-phase-compartment, which are not deactivated during a simulation. BioReservoirs that may be participate in two different sequential time compartments are preferably located and contained within the G0-phase-compartment, to maintain the continuity and to carry over the values of its concentration / density / scaled-amount from one sequential discrete time compartment to the next. These temporal compartments represent somewhat defined cellular phases with different activities.
- Mechanisms that are well studied in one animal species can be interpolated in the pathways of other species where the pathway may be less understood, such as in humans, by cloning the submodel containing the bioReservoirs and bioProcesses from the first species and transferring to the appropriate compartment of the second, and then renaming and reestablishing the appropriate connections. For instance, the genes that participate in programmed cell death or apoptosis in the nematode C. elegans have been identified, and it appears that *ced*-9 acts by inhibiting *ced*-3 or *ced*-4 activity. Since the human bcl-2 is also active when inserted into C. elegans (Vaux et al., Science 258: 1955) and it is also believed to be the homologue of ced-9, a model equivalent for that for the interactions of ced-9 with ced-4 or ced-4 can be also constructed for bcl-2 with ced-3-like or ced-4-like.
- BioPools that represent points of entry for non-physiological substances, such as drugs or synthetic agonists, antagonists or inhibitors that are not produced within the cells, at the compartment where it is first introduced have the following characteristics: a) they receive their values only from external sources, such as a simulation-panel, initially, the concentration equals the external input, and therefore Σ inputs = entry Value; b) they have a basalConcentration value of 0, c) they have no schematically modeled inputs, d) e) they may have several schematically modeled outputs, through modification- and translocation-bioProcesses, in addition to the decay term that represents diffusion, degradation and dilution, and e) the quantity variables of those bioPools cannot obtain values higher than those inputted by the user, but may get smaller over time due to the outputs and the decay factor ($[As]*(\mu+\Sigma \text{ outputs}) \leq |\text{entry}|$). When that compound is modified or translocated to another compartment, the latter's is a regular bioPool, but still constrained.

• **Modeling Cell Population Dynamics**

- There is a number of ways of modeling and simulating cell population dynamics, using this invention. A number of attributes defined for the classes cellReservoir, cellPool, cell bioModel, and other bioObjects can be used to hold parameters and variables to be used for that purposes. Following the teachings of this invention, other attributes can be similarly defined and used to better meet any particular biological system to be modeled, or hypothesis that a user may want to test, or to meet the needs for any monitoring or control system that use those biological models. Some examples are presented below.
- The state transitions are represented in the system of this invention by translocation bioProcesses between two bioPools, each containing entities in the former and in the latter states, at a rate that is determined with one of three approaches: a) mechanistic approach; b)probabilistic approach; or c) deterministic approach.
- In the mechanistic approach, the rate of the translocation process will be dependent on the dynamically changing values of certain variable or combination of variables, implicitly modeled inside an indicated cell-bioModel, reaching some threshold value(s). With this approach, the high-level models may be integrated with the mechanistic models, representing both the spatio-temporal integration in a single cell-bioModel and the quantitative data about populations of such cell modeled. This mechanistic approach is currently difficult to implement for the particular domain selected for this invention, since there is not sufficient information available. This approach can be implemented with better known systems in other domains.

- For some simplified in vitro biological systems, such as oocyte extracts, a mechanistic approach is used to model a transition which mechanism is known or expected, such as the role of the interplay between cyclin-B2 and the phosphorylated and dephosphorylated forms of the cdc2-kinase in the transition from the G2 to the mitosis phases of the cell cycle. The rate of increase in the number of cells with a cdc2**cyclin (cc)level greater that σ in a given cell cycle phase = fraction of cells in that phase (G_1) * [the rate at which cells within that phase reach a cdc2**cyclin level ($[cc]^{G1}$) greater that σ - the rate at which cells with cdc2**cyclin level greater that σ leaves that phase]:
- The following is an example of how a simplified mechanistic approach can be implemented. The rates of growth in eukaryotes and the rates of translocation from one cell-state-pool to the next can be represented as a linear or non-linear functions of the concentrations of specific molecules, such as external regulatory factors, regulatory enzymes, and total and specific mRNAs. For example, if a limiting protein L determines the transition from one time-compartment to another, then the expression of the mRNA for L could regulate that transition. Furthermore, the growth-rate μ can be represented as a linear function of the concentration of a given specific mRNA , and the rate of mRNA transcription can be represented as a non-linear function of the concentrations of two or more growth factors (or other regulators), or it can be assumed to be a combination of growth associated and stimulus-dependent kinetics. In turn, the steady state values of all enzyme synthesis rates and the total specific RNA level are a function (linear for RNA) of the steady-state specific growth-rate μ. The total RNA can be used as a measure of the growth resources of the cells, since the ability to increase a stimulus-induced specific sRNA is limited by the amount of synthesis proteins, which in turn is limited by the amount of total RNA. Because the relatively high rate of RNA turnover and the dilution effect of cell growth, the [sRNA] responds immediately to a removal of the stimulus-effector. With an increase of the stimulus-effector, the rate of increase of [sRNA] is controlled by a time constant which is inversely proportional to the total RNA of the cell (Copella &Dhurjati 1990. Biotech. Bioeng. 35:356-374). With the system of this invention, the modeler would clone from the Palettes set of bioReservoirs and bioProcesses of the appropriate types o represent the different terms of the following equations, which are then transferred to the appropriate compartments, connected and configured, and desired values are given to the model parameters. The following equations have to be adapted for use in the simulation formulas of the system of this invention. ($\mu = k_x * ([mRNA] - [mRNA]o) \rightarrow (1/h) = (gX/gRNA) * (gRNA/gX/h)$; $\tau = k_x / (\mu + k_x * [mRNA]o/[X])$, where k_x is the rate constant for mRNA transcription (gRNA/gX), τ is the time constant (gX/gRNA), and [mRNA]o/[X] is the specific [mRNA] at μ=0. The growth rate can also be viewed as the sum of all rates in one simple anabolic pathway divided by the total mass (growth from glucose in bacteria): $\mu = (\Sigma r_i)/[X]$. The number of generations: $\Phi = \log(2)((X - Xo)/Xo))$
- $\mu * G_1 * e^{\mu t} * \int_0^{\infty} \lambda^{G1}(x)dx = G_1 * e^{\mu t} * ([cc]^{G1}(\sigma) * \lambda^{G1}(\sigma) - r(G_1 \rightarrow S) * \int_0^{\infty} \theta(x)dx)$, where $\lambda^{G1}(x)$ and $\theta(x)$ are the frequency functions describing the cc content distribution for cells in G_1 and $G_1 \rightarrow S$ transition, respectively, and $r(G_1 \rightarrow S)$ is the rate constant that describe that transition: $r(G_1 \rightarrow S) = \mu / G_1 * (N_0 + G_2 + S)$; $r(S \rightarrow G_2) = \mu / S * (N_0 + G_2)$; $r(G_2 \rightarrow G_1) = \mu / G_2 * N_0$
- The following equations are examples of how overall rates of synthesis can be used to implement black-boxes, where each of those equations provide the value for the Velocity of the black-boxes to be represented by the corresponding type of bioProcesses: for IFN-α -> IL2 : $rs(TF\text{-}IFN\text{-}\alpha) = k_1 * [IFN\text{-}\alpha]$ and $rs(mRNA\text{-}IL2) = k_2 * [TF\text{-}IFN\text{-}\alpha]$ (or $rs(mRNA\text{-}IL2) = 1 + k_a [IFN\text{-}\alpha] / 1 + k_a [IFN\text{-}\alpha] + k_b$); $rs(IL2) = k_3 * [mRNAIL2]$.
- By using the concentrations variables of sol-mol-pools, expressed as moles per volume units rather that per cell, it would not be necessary to adjust the steady-state concentration with growth for constitutive enzymes, since it is assumed that both increase at the same rate, while for regulated enzymes, the dilution effect is more apparent, since it is not paralleled by synthesis. In the current implementation focusing on regulatory processes, since most of them take place in the form of large molecular complexes, highly localized in some parts of the cell, it would be more meaningful to use the defined density variables (molecules per cell). To synchronize the simulation from the intracellular to the cellPool level, the quantities of all the bioPools of extracellular molecules are halved at each cell division transition. In addition, specific rates are rates per cell, and total rates are specific rates multiplied by the cell density. Specific cell components may increase throughout the cell cycle: a) continuously at a constant rate, b) at a rate that doubles when the DNA is replicated, c) at a progressively increasing rate which follows fist-order kinetics, or d) discontinuously, according to a sequence of events which changes the rates in a complex pattern.

- When the mechanisms are unknown, a deterministic approach may be taken with the transitions modeled as a time-dependent events. As shown in Fig 2, the cell-phase compartments, which are holders of mechanistic models, also have an attribute "interval" to hold information about the experimentally observed duration of that phase of the cycle, which can be used during a simulation-run to deterministically deactivate such subworkspace when that period of time has elapsed since the subworkspace was activated. or whenever any type of known constraints are met. In the deterministic approach, the rate of the translocation process is defined by a rate constant, which may have a constant predefined value, or its value may be variable and set at run-time by a simulation formula, an expert rule, or procedure, such as one that reads the phase's "interval" just described, or it can be modeled by a model-block. The transfer of cells from cellPool i to cellPool j does not happen at once, but rather follows a Gaussian distribution, which may be modeled using the corresponding model-block.

- Following is an example of a high-level representation of the effect of some drug on cell growth, when there is partial mechanistic knowledge available, but where most of the many other steps in the pathways leading from the external drug to the regulation of cell growth are unknown. In a system where enzyme X_1 produces a soluble substance c which is used in the formation of active enzyme X_2. If X_1 and X_2 represent the total amounts of X_1 and X_2 contained in n cells, and if X_2 is the component which, upon reaching a threshold, determines the occurrence of cell division, so that $n = \beta X_2$, where β is a constant, then, assuming steady-state, when c remains at a steady concentration and therefore the formation of X_2 is proportional to $[X_1]$), follows that: $d[c]/dt = A(X_1/n) - B(X_2/n)[c] - C[c] = 0$, (1) where A, B, and C are constants. From (1) it can be said that $c = \beta X_1/X_2$, and following a set of derivations it results that the growth rate is: $k = d(\ln n)/dt = d(\ln X_2)/dt$. If now c decreases from the action of some drug, then X_2 which is proportional to [c] decreases. The cell growth rate $dn/dt = \beta\, dX_2/dt$ will also decrease. The effect of the drop in c is an increase in X_1 relative to X_2, until a new level of X_1 is reached that restores the cell growth rate to its initial value. In the representation used in this invention:
  - the 1st term is represented by an enzyme-bioProcess corresponding to the formation of c by X_1, where the enzyme-bioReactant would be connected to the bioPool of X_1, the substrate-bioReactant would be connected to the bioPool of the precursor of c, and the bioProduct would be connected to the bioPool of c;
  - the 2nd term is represented by an complex-formation-bioProcess corresponding to the consumption of c in the formation of X_2 from c and other subunit(s), connected to the appropriate bioReservoirs; and
  - the 3rd term could be represented, depending on the specific biologic mechanism to be modeled, by: a) a complex-formation-bioProcess corresponding to the consumption of c in the formation of a complex leading to the loss of c by degradation by a ubiquitin-dependent mechanism, connected to the appropriate bioReservoirs; or b) a translocation-bioProcess, representing other mechanisms of loss of c, such as by diffusion, connected to the appropriate bioReservoirs; or c) implicitly represented by the degradation-coeff within the formula that determines the current amount of c, which includes a term with default degradation.
  - in this system, the scaled-amount set of variables is used, because what is relevant for the behavior of the system are the relative value of X_2 in relation to its threshold, which determines cell division, and the ratio between X_1/X_2, which regulates the rate of production of X_2, and not their absolute amounts.
  - the value of the number-of-cells attribute of the cell-bioModel that contain those bioProcesses and their corresponding bioReservoirs, is doubled when the scaled-amount of the bioPool of X_2 reaches the threshold, and at the same time, X_1, X_2 are halved while [c] remains constant.
  - The above example is an oversimplification, and in fact much more is currently known about the mechanisms that determine the initiation of mitosis, in which complex interdependencies are involved, such as the equivalent of X_2, a complex, regulating not only the rapid increase in its production by a positive feedback loop upon the equivalent of X_1, but also X_2 regulating its destruction by activating an enzyme in the ubiquitin-dependent mechanism, and X_2 regulating its inactivation by activating another enzyme that converts X_1 back to c. In general, in the absence of diffusion, the simple case that might produce organization in time is that of two concentration variables, X_1 and X_1, where the value of each is dependent on the value of both, such as when: $d[X_1]/dt = v_1(X_1,X_2)$; and $d[X_2]/dt = v_2(X_1,X_2)$.
  - Other high-level physiological events can be modeled in a similar manner, such as considering different cells subsets (i.e. neutrophils) changing compartments (intravascular -> interstitial space during inflammation) as induced by cytokines (i.e. TNF-α, which is produced upon activation by macrophages, neutrophils, T-lymphocytes, natural killer cells and mast cells), in chemotactic-bioProcesses. The time-period that a cell subset stays in such a compartment depends in part on the expression of adhesion proteins (such as E-selectin

ELAM-1) or ICAM-1 by endothelial cells, which expression is induced by TNF-α), that will be components of cell-interaction-bioProcesses.

- **Modeling Progression through Different Cell States: The Cell Cycle and Cell Differentiation**

- Each cell-phase has two attributes called the Generation-number (Φ) and the Number-of-cells (N, per volume unit). At each cell division at the mechanistic level, that is, when the subworkspace of the M-phase compartment is deactivated and optionally the subworkspace of the G1-phase compartment is activated, predefined exper rules set $\Phi = \Phi + 1$ and $N=2n$ for each cell-phase of that cell-bioModel. Another attributes called Mass-per-cell (X), can optionally be given a value and used to calculate specific values as a function of cell mass, in which case predefined expert rules (Table 78) set $X = X/2$ for the Go-phase, and the G1-phase if applicable, of that cell-bioModel, and then the value of that attribute would be modeled as dependent on time or as dependent on any other variable(s), such as the concentration of some specific mRNA(s) or protein(s), until reaching again a value of X or close to X. In turn, the value of X can be used to adjust other quantities or to control the activation and deactivation of the subworkspaces of each successive cell-phase of that cell-bioModel, making it dependent on cell-size. The equivalent at the population level at mitosis is that n cells from the M-phase bioPool are translocated to the G0-phase (or the G1-phase depending on the situation) bioPool while multiplied by 2, so that the G0(G1)-phase bioPool receives 2n cells.
- FIG.13 is a representation of the mechanistic approach, and attempts to represent on paper what can be achieved dynamically at run-time applying the innovations of this invention. If we assume that a cell is in Go-phase state and focus on that state's sub-workspace (1301), and there we assume that variables represented as Z and W have received the values z and w, respectively, then following the implementation described in this invention, a rule will be invoked (1102) that will activate the subworkspace of the G1.1-phase (1303) compartment. Activation of G1.1-phase.sw implies that all the variables and parameters encapsulated within the bioObjects contained upon that subworkspace will be available to the simulator, and the data flow can now continue through any branch of the pathways represented there. Therefore, upon required events being satisfied, such as Z and W reaching certain thresholds -which could be an effect of binding one or more growth factors to their receptors on the cell membrane-, the cell makes a transition from the G0 to the G1.1 phase. The subworkspace of the G1.1-phase remains activated during the whole simulation run because it contains all those bioProcesses that are not cell cycle specific. Focusing now on the simulation of the G1.1-phase (1303), two expert rules could be invoked, saying that "Whenever variable A receives a value $a > \alpha 1$ then activate G1.2-phase.sw" (1304), or "If X and Y receive values $x > \alpha 2$ and $y > \alpha 3$ then activate Differentiated-Cell.sw" (1305) and depending on which one is invoked first, either the subworkspace of the G1.2-phase (1306) compartment or the subworkspace of the Differentiated-Cell.sw (1307) compartment will be activated, since the subworkspace of the G1.1-phase (1306) compartment will be deactivated by another two expert rules that state that "Whenever the subworkspace of the G1.2-phase (or Differentiated-Cell) is activated then deactivate the subworkspace of the G1.1-phase". This way of reasoning apply to the other cell-phases as well, with the difference that the options vary for each cell-phase. For example, the G1.2-phase (1306) and S-phase (1308) may be committed to only one option, with the time for progression to the next phase being determined by one rule, as shown, while the G2-phase (1309) and M-phase (1310) have two options, and which one will be activated is controlled by two rules, as described above. One of the options, Apoptosis (1311) is a dead-end stage, representing regulated cell death. While in M-phase (1310), a live cell's program tells it, before or during cell division, to continue proliferating, in which case one or both the daughter cells entering again the G1.1-phase, by activating the subworkspace of the G1.1-phase, or to exit the cell-cycle, in which case the simulation continues with only the Go-phase. Note that the options for each one of the cell-phases have been predefined by the characteristic number of stubs for each subclass .
- The dynamics system diagram in that figure is used to represent that the defined attributes of the subclasses of cell-phase contain a variable that allows to keep track of the proportion of the cells, from the original number on Go-phase, are progressing through the phases of the cycle, as an alternative to build a model with a set of bioReservoirs and bioProcesses. This variable can be modeled using a probabilistic or mechanistic approach. In a probabilistic approach for representing cell-phase distributions, a balanced growth type of cell population is characterized by such distributions, which is a time invariant property even if their total number increase exponentially with a constant specific growth rate μ. Each of the subsets of that population, which represent different cell-phases, and which can also be represented as separate cellPools in the system of this invention, increase in number at the same specific rate and are also characterized by time invariant property distributions, which is represented by the "typical-fraction" attribute of each of the cell-phases. At some steps during a simulation, the alternative cell-phase compartments that could follow may exclude each other at the single cell

level, but not at the cell population level, so that different subsets of a population may follow different paths, with the distribution depending on the strength of signaling events generated within the simulation itself. Therefore, the simulation of the value of this variable could be modeled to represent the probabilistic distributions, independently of which subworkspaces are activated first following the mechanistic approach

- As shown in FIG.32, If a cellular response results only in proliferation, the simulation of the population cell dynamics involving bioPools of cells in different states, results in cells that keep translocating from bioPool to bioPool, each representing a different one from a discrete number of states, resulting in cycles through a closed-loop set of cell-state bioPools, one of those . However, if the response results in cellular differentiation, it generates branching to a new closed-loop set of cell-state-pools.
- FIG.32 shows how a cell populations approach bioModel can be build to model the translocation of cells between the major states considered, as mechanistically represented by the defined subclasses of cell-phase, by creating and connecting a set of bioReservoirs and bioProcesses, as the example detailed in the figure shows. Although the cell cycle is a continuous one, it can be compartmentalized into functional cycle intervals whenever it is deemed useful to better represent the dynamic simulation of the system. The initial state is the resting state, represented by the cellPool Cell-X1-Go (3201), which contains cells of type X1 (or in differentiation stage X1) in the Go-phase. That cellPool may receive inputs (3202) from various sources, represented by bioProcesses in which cells are added to the resting pool after having been activated or cycling as the result of a variety of factors, and its outputs (3203) may contribute also to a variety of such bioProcesses, such as the one that will be here described. While at Go-phase, if the cell is activated (3204) by a growth factor (3205), antigen or any other activating stimulus, the cell enters the activated state or G1.1-phase (3206) and added to the corresponding bioPool (3207). From that bioPool (3207), all the cells may follow one of the two different options, or different cells may follow different options for the next cell-phase, depending on the type and strength of the signals received, either: a) following the cell cycle (3208) if certain growth factor(s) (3209) are sufficiently present, resulting in cells X1 in G1.2-phase pathway (3210) and added to their corresponding bioPool (3211), or b) following the cell differentiation pathway (3212) if certain differentiation factor(s) (3213) are sufficiently present, resulting in cells X2 in G1.1-phase (3214) and added to their corresponding bioPool (3215), and these cells will now follow a new set of . Each of the cell types will be then running in parallel, with their own set of bioReservoirs and bioProcesses. The same reasoning applies to the rest of the system, as detailed in the figure, which will not be further explain, only to say that and the end of the cell cycle, in the cellEngine (3216) that returns cells from M-phase (3206) to G1.1-phase (3206), has not only a translocation function, but in addition, given that the "stoichiometric-coeff" of cell-X1-Go is 2 while the "stoichiometric-coeff" of cell-X1-M is 1, it doubles the number of cells to be added to the cell-X1-Go-pool.
- In fast growing organisms, the dilution factor resulting from cell growth is represented by μ, while in slow growing cells direct enzyme degradation is more important, since the levels of many enzymes in animal cells is controlled by the balance between enzyme synthesis and degradation, called turnover. The synthesis of an enzyme is a zero-order process, while the degradation usually has a first-order kinetics (proportional to the concentration of enzymes.
- It is very difficult to measure directly the rates of change of different cellular components at the single cell level A modeling framework requires at least data from measurements about the distributions of single-cell components in growing cell populations, and at the time of cell division and birth, which in some cases can be obtained by multiparameter flow cytometry.
- Growth rate in terms of net rate of protein accumulation in single cells in each phase of the cell cycle.
- The growth rate in terms of **net rate of protein accumulation in single cells in each phase of the cell cycle** can be calculated, for balanced growth cell-populations, based on the measured distributions of single-cell protein content in the total cell population. Similarly, net rates of accumulation of any specific protein as well as any other compound or property can be calculated whenever the properties and their distributions can be measured on a single cell basis, such as with multiparameter flow cytometry with gating features, that permit direct correlation with the cell-cycle phase. Kromenaker and Srienc (1991) **defined balanced growth-cell populations** as those characterized by: a) the total cell number increasing exponentially with time according to a specific constant mean growth rate, and b) the total cell population and any of its sub-populations having a time invariant single-cell property distributions. They developed equations to represent the dynamics of a given protein content range in an individual cell cycle, and after deriving restrictions that apply to the rate constants and substituting those in the original equations, the obtained expressions for the growth rates for each individual cell cycle. The transformation of this model into the graphic language proposed in this invention, and the incorporation of those equations, is an example of a currently preferred embodiment of this invention, which is explained in more detail in Exhibit M1.

- The transition between different time-compartments may also be made a function of cell size, or its equivalent total-protein content, both of which are attributes of cell-bioModel. At the population level, the transfer of cell-units from the bioPool of cells in G1.1-phase to the bioPool of cells in G1.1-phase is also based on the value of such attributes. For example: the rate of increase in the number of cells with a protein content greater than a given value p in G1, which may be established to delimit the G1.1-phase from the G1.2-phase compartments, equals the rate at which cells with a protein content greater than p enter G1-phase + the rate at which cells within G1-phase grow into the class of cells with a protein content greater than p - the rate at which cells with a protein content greater than p exit the G1-phase.
- Additional sub-phases can be analyzed whenever appropriate landmarks can be observed to define additional transitions. To represent a specific form of cell death, which is highly regulated and known as programmed cell death or apoptosis, a separate time-compartment is defined in this system, which is equivalent to a cell-cycle phase compartment but with the difference that has a dead end and does lead to a next compartment, but rather eliminates cells that have passed through certain specific combination of events. In addition, an additional term representing the rate of exit from each cell-cycle compartment via other types of cell death may be introduced into each of the population balance equations.
- In this system, in general it is assumed that cell division is symmetric, and that total protein is conserved at cell division. The G1-phase is bounded by cell birth and initiation of DNA replication; S-phase is bounded by the initiation and completion of DNA; G2-phase is bounded by the completion of DNA synthesis and the initiation of mitosis; and the M-phase is bounded by the initiation of mitosis and cell division. At balanced growth, each subset of cells in each cell cycle phase increase in numbers at the same specific growth rate as the overall population. The rate of entry into the G1 phase is twice the rate of exit from the M phase. Additional phase compartments can be added whenever other transition landmarks can be measured, as well as the cell distribution at each side of the transition. The following can be used as the equations to model the rates: $r_{G2G1} = k * No / G2o$, $r_{G1S} = k * (No + G2o + So) / G1o$, $r_{SG2} = k * (No + G2o) / So$, where No is the number of cell in the total population at some reference time, and $No = G1o + G2o + So$ at this time
- The currently preferred embodiment of this invention deals with different stages of cell differentiation, each representative of a characteristic set of behaviors, as separate pools of cells. All the cells in a pool have identical behavior and, again, in the system of this invention when one refers to an instance of a cell with a particular phenotype it does not mean an individual cell but rather a pool of equal cells that is characterized by its behavior and also by a cell-number and a cell-density. We have also considered previously cases where individual cells grow and differentiate in response to specific signals provided when environmental factors activate specific receptors that initiate a signal transduction pathway within the cell. However, different cells respond by following diverse pathways in response to an identical signal, and this specific response depends on the history of that cell. For example, erythroid cells could be allotted to three pools with phenotypes characterized by the presence of two cell differentiation surface markers CD34 (a glycoprotein) and CD71 (the transferrin receptor): a) $CD34^+$ include most immature erythroid progenitor cells; b) $CD71^{low}$ includes intermediate stages; and c) $CD34^-CD71^{high}$ includes most mature erythroid progenitors. Each of this populations have different predominant form of the erytropoietin receptor, and respond in different ways to erytropoietin: $CD34^+$ express a truncated form of the receptor and respond by proliferation, while $CD34^-CD71^{high}$ respond by differentiation and are protected from apoptosis or programmed cell death (Nakamura et al., Science 257, 1138-1141, 1992). In the currently preferred embodiment of this invention, the particular history of each cell pool is reflected in the specific bioModels as determined by the particular procedures to be followed, which are encoded in the corresponding attributes of the bioObjects that are components of that cell's specific bioModels.

3. **Navigation and General Modes: Exploring and Queering the Virtual Models**

a) **Menus and Panels**

- The domain-menus associated with the Navigation Mode and the General Mode, the default user mode for users with no modeler or developer rights, comprise the headings: "Help & Mode Menus", "Structure Libraries", "Pathway Libraries", previously discussed, and "Panels", which options include panels of various types used to access tasks that are not initiated from one single bioReservoir or bioProcess, or their components, but which are either application-wide or including more that one starting points.
- One of the panels is the Initialization Panel is used to request the initialization tasks associated with each of the buttons. Here we will discuss the methods associated with each of those buttons. An initially rule (Table 97) causes this panel to be automatically displayed when the program is started. Selecting the "Menu" button calls the basis-menu-head-callback (Table 98) which causes the creation of the specific domain-menus, by calling other procedures defined within the Shell and, depending on the current value of the user-mode of the

window from which the request has been made, it passes as argument the name the menu-structures specific for that mode. A menu-system-template is a facility provided by the Shell that allows to build permanent designs of connected structures or menu-templates of different classes, each corresponding to a different menu option, and each containing attributes that hold information about the "label" to be displayed by such menu option, the name of the procedure to be invoked, and depending on the class of such menu object, it may hold either the name of the object which subworkspace is is to be displayed, or the name of a new menu system to be displayed. When an option is selected from the menu, the procedure named in its corresponding template is started, which may take as arguments the values of some of those other attributes. The domain-menus are transient sets of workspaces and messages that are dynamically created mirroring the ordered design of the named permanent menu-structure.

- Selecting "References Scroll" displays the the subworkspace of the References Panel, discussed below. Selecting "Structure Queries" creates a clone of Master-General-Structure-Query-Panel and displays its subworkspace discussed in more detail below in relation to FIG.30. Selecting any of the "Experiment Setup #" displays the subworkspace of the corresponding "Experiment Setup Panel", which are variously modified clones of the Master-Experiment-Setup-Panel, one of which will be discussed in more detail under the Simulation Mode heading in relation to FIG.35, since part of the capabilities provided within this type of panels can be used under the General Mode, and all the capabilities can be used in Simulation Mode.

b) **Initialization**

- Selecting the "**Navig Init**" button calls the navig-init-callback, which in turn calls the navig-init-proc (Table 99), which results in the activation of the subworkspaces of all bioProcesses and bioReservoirs (which are all deactivated when the application is initially started), and then successively calling the following procedures:
  - the br-initialization-proc (Table 100) sets the appropriate values of the "master-bioReservoir" attribute of all bioReservoirs, highlights the flag-color region of the icon of those bioReservoirs that have "warnings", sets the appropriate values of the "ref-bioprocess" attribute of all bioPool-posts of all bioReservoirs, and establishes all the appropriate "a-down-bioProcess-of" and "an-up-bioProcess-of" relations between bioProcesses and bioReservoirs, making then those changes permanent (note that relations are transient and cannot be made permanent, and therefore they are lost when the application is restarted).
  - the bp-initialization-proc (Table101) changes the color of the type-color region of the icon of each bioProcess to equal that of the bioEngine it contains, sets the appropriate values of the "master-bioprocess" attribute of each bioProcess, highlights the flag-color region of the icon of those bioProcesses that have "warnings", sets the appropriate values of the "ref-bioreservoir" attribute of all bioRole-posts of each bioProcess, establishes "an-upstream-bioReservoir-of" and "a-downstream-bioReservoir-of" relations between certain pairs of bioReservoirs that are separated only by one bioProcess, and establishes "an-upstream-bioProcess-of" and "a-downstream-bioProcess-of" relations between certain pairs of bioProcesses that are separated only by one bioReservoir.
  - the downstream-chaining-proc (Table 102) completes the downstream chaining process by extending those relations to distant bioReservoirs and bioProcesses, by establishing the appropriate "a-downstream-bioReservoir-of" relations between any bioReservoir A and each other bioReservoir that is positioned downstream of A in any pathway, and establishes all the appropriate "a-downstream-bioProcess-of" relations between any bioProcesses B and each bioProcess that is positioned downstream of B in any pathway.
  - the upstream-chaining-proc (Table 103) completes the upstream chaining process by extending those relations to distant bioReservoirs and bioProcesses, by establishing the appropriate "an-upstream-bioReservoir-of" relations between any bioReservoir A and each other bioReservoir that is positioned upstream of A in any pathway, and establishes all the appropriate "an-upstream-bioProcess-of" relations between any bioProcesses B and each bioProcess that is positioned upstream of B in any pathway.
  - the biomodel-initialization-proc (Table 104) scans each bioModel for other contained bioModels, bioProcesses and bioReservoirs, going down the subworkspace hierarchy, and establishes the following relations between any combination of them: bm-contained-in between a bioModel and a bioModel, bp-contained-in between a bioProcess and a bioModel, and br-contained-in between a bioReservoir and a bioModel (Table 47). Those relations are important for all further processing, and are used by the newly defined query procedures, navigation procedures, and simulation procedures described below.
  - A rule (Table 119) monitors the application at run time, and whenever the user moves a bioModel the rule starts the biomodel-ws-check-proc, which checks whether the bioModel has been moved to a different workspace, in which case it calls the remove-biomodel-containing-proc (Table 119), to brake the old relations that no longer apply, and calls this biomodel-initialization-proc (Table 104) to establish the new relations. The upstream and downstream relations are also created and broken at run-time by a set of rules (Table 76) and a parallel set for

bioReservoirs, not shown) that monitor whenever any such chaining relation is newly established or broken, and in response to such events they establish or brake other relations, according to similar criteria, which in turn trigger the same rules for a chain of other bioReservoirs or bioProcesses. The use those rules makes sure that if a modeler or user deletes a bioReservoir or bioProcess that is part of one such chain of relations, then the chain is broken between the elements of both sides on the breaking point. Similar sets of rules are used for establishing relations whenever new connections between bioReservoirs and bioProcesses are made by the modeler.

- The "Add Query" button is to be selected only after the "Navig Init" was been successfully concluded. If changes to the database have been made by the modeler since the "Navig Init" was last concluded, or if the application has been restarted, then the "Query Init" button should be selected. The common purpose of both buttons is to organize knowledge about the structure of all the bioEntities in the application, and to create a set of structures to hold that knowledge, to be used any time that the user request a predefined query that involves that kind of knowledge. Selecting the "Add Query" button calls the add-query-callback (Table 105), which in turn calls the two procedures that follow. Selecting the "Query Init" button calls the query-init-callback (Table 106), which in turn calls the navig-init-proc (Table 99) and the two procedures that follow:
- the et-initialization-proc (Table 107) scans the loaded application for every copy of each named complex-bioEntity, and then assigns a value to the "id" attribute of each such copy that is equal to the name of such bioEntity complex-bioEntity ended by the copy number. Copies are numbered consecutively, in the order found, with the original being -0.
- the init-query-tables-proc (Table 108) activates the subworkspace of the Query-Arrays-Bin and deletes any old array upon it, and also deletes any old list upon the Query-Lists-Bin, and then call the following procedures:
- the find-bioentity-definitions-proc (Table 109) scans the Bioentities-Defs-Bin, which contains all the object definitions of all the subclasses of the class bioEntity, and for each of those subclasses it calls the create-query-array-proc (Table 110) which creates a query-array, sets its "major-class" attribute equal to the name of the superior-class of such subclass, sets its "ref-component-class" attribute equal to the name of such subclass, and sets its "name" attribute equal to the name of such subclass followed by the ending -query-array, and transfers such query-array to the Query-Arrays-Bin at a location specific for each superior-class;
- the create-query-lists-proc (Table 111) creates a query-list to correspond to each query-array in the Query-Arrays-Bin, and transfers the list to the corresponding location upon the Query-Lists-Bin, sets its "ref-array" attribute equal to the name of such array, and sets its "ref-component-class" attribute equal to the "ref-component-class" attribute of such array;
- then, for each of those lists and for each bioEntity of the subclass named by the "ref-component-class" of such list, it calls the find-named-bioentities-with-component-proc (Table112) which in turn, if such bioEntity has a name it directly calls for such bioEntity the procedure that follows, otherwise it calls first the find-named-bioentity-proc (Table 113) to scan the hierarchy of bioEntities superior to such bioEntity, until one is found with a name, in which case, it now calls for the named bioEntity the procedure that follows:
- the seek-bioentity-copies-proc (Table 114), inserts the argument bioEntity in such list, and then scans the application for any copy of such bioEntity, and if any is found then it loops back for such bioEntity by calling the find-named-bioentities-with-component-proc (Table 112) mentioned above, which will repeat the cycle back to this procedure, as described, until the various loops for each bioEntity of every list are completed.
- When this process is completed, the list for each bioEntity subclass contains, with no duplicated elements, all the named bioEntities that contain one such bioEntity, either directly or by reference. To give an example, that means that if a named protein instance-A contains a protein-motif instance-B, then every named protein complex-instance-C that contains an unnamed copy of instance-A (which does not itself contain a copy of instance-B, but rather points to its master-bioEntity, instance-A, which contains instance-B) will also be included in the list of all bioEntities that contain a protein-motif. After all such lists are completed, it calls the following procedures:
- the complete-query-table-proc (Table 115) simply transfer the values of the elements from each of those lists to their corresponding arrays, by fist setting the length of the array to match the the length of the list, and then setting the text of the initial-values attribute of each array to equal a text string that mentions all elements of the list separated by commas, as illustrated in the example listed in Table 116, which shows a pair of corresponding instances of a query-list and a query-array of the subclass sup-et-array, which list named instances of bioEntities that contain an a-helix-motif, which in this small demo application consists only of a few master-structures from the library. In the table of attributes listed, the initial-values attributes is displayed, and the list of elements of either the list or the array can be displayed by selecting the "describe" option of their menus.
- the create-auxiliary-query-arrays-proc (Table 117) adds a few empty arrays corresponding to the no-selection options of each query panel, which are simply programming aids

- and to conclude it deactivates and activates again the subworkspace of the Query-Arrays-Bin for the purpose of initializing the query-arrays upon activation, which means that the elements of the array take the values given by the set initial values.
- The "**Icons Init**" button calls the init-icons-callback (Table 118), which function is not essential for the functioning of this system, but which provides a very useful cosmetic function, by realigning the tracers in the subworkspaces of bioReservoirs and bioProcesses and making their connections invisible. Selection of this button is not necessary every time the application is started, and it can be selected any time thereafter, when problems are detected, in any mode other than Simulation Mode (where those subworkspaces may be deactivated). The same effect is achieved for individual bioReservoirs or bioProcesses by slightly moving and repositioning their bioPool or bioEngine, respectively, which triggers their respective icon-movement rules, as defined in Table 79, which also apply to repositioning the bioRole-posts any time a bioReactant or bioProduct is moved.
- Selecting the "**Init All**" button calls all-init-callback (Table 118b), which in turn calls the all the initialization procedures mentioned above, and selecting the "**Hide**" button calls the hide-workspace-callback, which hides this Panel.

c) **Navigation**

- In this invention, navigation refer to the domain-specific tasks of displaying and hiding either related bioObjects or their subworkspaces. The concept of related object include various types of associations, including some that are permanently stored in the knowledge base, as defined by either graphic or distant connections, or containment in subworkspaces at any level in the hierarchy, and some that are transient such as relations, which are established and may be broken at run-time (note that in the system of this invention it is also possible to store run-time versions or snap-shots of the application, which also include transient objects, such as relations). Navigation tasks may may be performed in any mode, but Navigation Mode has that name because the behavior of certain object classes, including the classes bioReservoir, bioProcess, and complex-bioEntity, is restricted to do just that. This means that a navigation task is automatically performed upon clicking on the instances of any such classes, instead of displaying a menu with various other options as well, and therefore all those other tasks offered by the other options are not available in this mode. Therefore, a user in Modeler or General Mode may want to quickly switch to Navigation Mode when intending to do just that, because it is much faster, for example when exploring and navigating through the pathways, and then switch back to the other mode to perform other tasks. Navigation tasks are usually simplified, in the following discussion and elsewhere in this filing, by expressions like "selection of A starts proc-B which displays C", but in reality these processes do more than that in this invention: selection of A invokes proc-B which takes as arguments A (the item) and the current window, and causes C to be either displayed or hidden on the current window, depending on the value of the "toggle-state" attribute of A, and in addition, the icon appearance of A is configured, switching between its pressed or depressed configuration, respectively, and the value of the "toggle-state" attribute of A is alternated between on and off. In such cases, A may be either a button or any of the bioObjects that has a subworkspace, which in this regard behave similarly.
- FIG.30 will be described to summarize some of those navigation tasks when in Navigation Mode, while also referring to the definitions of the methods involved listed in the Appendix. Note that any navigation task that is directly performed in this mode, may be performed in any other mode as well, but depending on the previously defined class restrictions for each mode, they may have to be selected from the menu instead, requiring two steps rather than one. Selecting any bioProcess in Navigation Mode implies selecting its "details" option which starts the details-bp-proc (Table 120) which displays its subworkspace, in the same way that selecting any bioReservoir implies selecting its "details" option which starts the details-br-proc (Table 121), resulting in the subworkspace of the bioReservoir being displayed or hidden, depending on whether the value of the "toggle-state" of the bioReservoir is hide or show, respectively. With both procedures, the icons of the tracers on the corresponding subworkspace are restored to their default relative positions before such subworkspace is displayed.
- Note that when selecting the "details" option of any bioProcess that has a master-bioprocess but has no subworkspace, conditions which when combined are characteristic of a copy bioProcess used in the creation of Pathways, causes such first procedure to call the bp-master-details-proc (Table 120) which in this case displays or hides the subworkspace of the original (master) bioProcess, in addition to configuring the icons of both the original and the copy bioProcess to their pressed or depressed appearance, respectively. In a similar way, when selecting the "details" option of any bioReservoir that has a master-bioReservoir but has no subworkspace, characteristic of a copy bioReservoir, causes the first procedure to call the br-master-details-proc (Table 121) which in this case displays or hides the subworkspace of the original bioReservoir, in addition to configuring the icons of both the original and the copy bioReservoirs to their pressed or depressed appearance, respectively

- BioEngines do not have subworkspaces by default, by the modeler can create one, as described, to store tabular functions. In such cases, a "show-sw" option is available, and selecting such bioEngines in Navigation Mode implies "show-sw", which starts the show-bioengine-sw-proc which displays or hides its subworkspace (not shown) and toggles the configuration of its icon.
- The behavior of complex-bioEntities in Navigation Mode is similar, as shown in FIG.18, where selecting any bioEntity (1808, 1813, 1815, 1817, 1819, or 1821) implies selecting the "details" option which starts the et-details-proc (Table 122) which displays or hides its subworkspace (1809, 1810, 1816, 1818, 1820, or 1822, respectively) and toggles the configuration of its icon. The combination of this option with the go-to-sup-obj-button (1825) upon each subworkspace allows to navigate up and down in their workspace hierarchies.
- Selecting any bioProduct-post (711) or any bioReactant-post (703), in General, Navigation, or Simulation Modes, implies selecting its "show-br" option which starts the rolepost-show-br-proc (Table 123) which displays the subworkspace (714 or 705) of the bioReservoir named by their "ref-bioreservoir" attribute. Similarly, selecting any bioPool-r-post (704) or any bioPool-p-post (712) implies selecting its "show-bp" option which starts the poolpost-show-bp-proc (Table 124) which displays the subworkspace (402) of the bioProcess named by its "ref-bioprocess" attribute. This option is only available in the menu when the bioPool-post has a name. Selecting the go-to-sup-obj-button (with triangle) on the subworkspaces of each bioObject displays its superior object, while selecting the hide-sup-ws-button (with cross) hides its workspace . The combination of those capabilities allow to follow all the distant connections through the pathways of bioReservoirs and bioProcesses, and also up and down in their workspace hierarchies.
- Selecting any model-box (511, 516, or 517) in General, Navigation, and Simulation Modes, implies selecting its "show-sw" option, available only if the model-box, which starts the show-model-box-sw-proc which displays or hides its subworkspace and toggles the configuration of its icon. This option is used to access the model-block upon that subworkspace.
- Other important navigation task performed through the menu in any mode, include the "bioentity" option (Table 125) appears in the menus of bioReservoirs, bioReactants and bioProducts only when the "ref-bioentity" attribute of the bioReservoir has a value different from the default value of none, and upon selection of this option by the user, or hides, depending on whether the initial value of the toggle-state attribute of the bioReservoir is hide or show, respectively, the subworkspace of the bioEntity named by said ref-bioentity attribute. Note that the "ref-bioentity" attribute is defined only for the class bioReservoir, for storage economy, and the menu options of bioReactants and bioProducts refer to that attribute when connected. However, this attribute can be defined for those other classes as well, allowing access to a bioEntity independently of a bioReservoir.

d) **Structure Related Queries**

- There are two major groups of Queries: a General-Query is based only on structure and is application wide, while a BR-Query is in reference to the bioReservoir from which it was initiated, to be described below in more detail. The Query Panels are built hierarchically, with each radio-button from the first Selection Panel leading to another more specific Query Panel, where now the user can compose a selection and then initiate the Query Note that the word Panel is frequently used in a more general sense to refer to the subworkspace of an instance of the class query-panel.
- A General-Query may be requested by selecting from the "Panels" menu head the "Structure Queries" option which calls the create-general-structure-selection-panel-callback (Table 127), which in turn clones the Master-General-Structure-Query-Panel, stores it in the Temporary-Panel-Bin, configures its radio-box with the aid of the set-radio-box-id-proc (Table 128) and displays its subworkspace. The user may now use such Panel to select any radio-button from its radio-box, which calls the select-general-structure-selection-panel-callback (Table 129) with such button as one of its arguments, which in turn: a) gets the value of the radio-button selected, and depending on the value of the "on-value" attribute of such button, clones the corresponding master-panel, then calls the create-general-query-panel-proc (Table 130) with such value and the new panel among its arguments, which in turn completes such panel by scanning the panel for the presence of specifically labeled radio-boxes (related sets of radio-buttons) and configuring the radio-boxes with the set-radio-box-id-proc (Table 128), and b) displays the subworkspace of the new panel.
- The user may now use that Query Panel to select any combination of one radio-button per radio-box and then selecting the "Query" button calls the query-general-structure-callback (Table 131), which in turn calls first the query-general-structure-proc (Table 132), which scans such panel for the presence of specifically labeled radio-boxes and then calls find-specific-array-proc (Table 133), with such label as one of its arguments, to find the value of the selected radio-button and the matching array within the Query-Arrays-Bin (those arrays are

created during the Query Initialization process, as described in the Initialization section above). An auxiliary scroll-text-list is created and, if only one radio-box is present in the Query Panel, then the elements of the corresponding array are transferred to such list. If two radio-boxes are present in the Query Panel, then two such arrays should be found, in which case the merge-two-structure-lists-proc (Table 134) is called, which scans those arrays to find the common elements, which are transferred to such scroll-text-list. An specific text-string is developed to include references to the selected options and to the outcome of the search, which is then given as a value to a free-text structure which is displayed as a heading for the results of the search. The callback calls then the create-general-query-output-panel-proc (Table 135) with such scroll-text-list and free-text among its arguments, which creates an output panel by cloning the Master-Molecular-Query-Output-Panel, then creates a scroll-area to display the elements of such scroll-text-list, and transfers to the subworkspace of such output panel both the free-text and the scroll-area listing the results of the search, which is them displayed. The user can now use such scroll-area to interactively display or hide the structure of any number of bioEntities named by the options listed, as described below.

e) **Pathway Related Queries**

- FIG.30 illustrates an example of one of the several types of predefined Pathway Related Queries that the user can perform by just selecting the desired buttons from the options offered in the panels. All the searches generated from this type of panels include bioReservoirs located anywhere within the pathways that are either upstream or downstream, depending on the selection, of such initial bioReservoir, and also the bioEntities referred to by any of those bioReservoirs. Those upstream or downstream queries can be further restricted by the type of roles that those bioReservoirs play in different bioProcesses, as defined by the major classes of bioReactants to which their bioPools are distantly connected, which is referred here as function-related search In complex queries of mixed type, such bioEntities can be further restricted to those containing any particular component, similar to the Structure Related Queries described above. In such cases, the methods pertinent to the pathway search are applied first to generate the pathway-related set, followed by the function search, which is applied only to the pathway-related set and generates the function-related set, and finally the structure search, which is applied only to the function-related set, or to the pathway-related set if no function search has been selected, and generates the structure-related set. The structure search applies only to the bioEntities for the purpose of displaying the results, in the sense that all bioReservoirs that meet the selected pathway, and optionally function, requirements are listed in the scroll-area listing the bioReservoirs, regardless of whether they meet any of the bioEntity related criteria. This implementation is currently preferred because the "ref-bioentity" of a bioReservoir is an optional attribute, and it is expected that many of the bioReservoirs will not have a corresponding defined bioEntity. This alternative provides a more complete search in terms of relative position and function of bioReservoirs, without being influenced by the implementation of an option.
- These types of Query may be requested in two different ways: a) from a bioReservoir (3001) by selecting from its menu (3002) the "query panels" option (3003); or b) from the subworkspace a bioReservoir by selecting the query-tracer (1325) connected to the bioPool. A **query-tracer** is a subclass of list-tracer (Table 30) which selection implies selecting the tracer's "show" option. Each of those options starts the create-br-related-selection-panel-callback (Table 138), which creates a clone of Master-BR-Molecular-Query-Panel, configures it, and displays its subworkspace (3004), the Pathway Query Selection Panel. Configuring this panel and any other panel that follows during the processing of br-related queries always includes to set the value of the "related-item" attribute of the panel equal to the name of the bioReservoir from which the query was initiated, and this name is passed along when a new panel is originated from such panel. That is, any callback invoked from a button of a panel, takes the name of the bioReservoir from the "related-item" of the panel upon which subworkspace it is located, and passes that name as an argument when calling other procedures.
- The user may now use that Pathway Query Selection Panel to select any radio-button (3005) from its radio-box, which calls the select-br-related-query-panel-callback (Table 139) with such button as one of its arguments, which in turn: a) gets the value of the radio-button selected, and depending on the value of the "on-value" attribute of such button, clones the corresponding master-panel, then calls the set-br-related-query-panel-proc (Table 140) with such value and the new panel among its arguments, which in turn configures and completes such panel with the set-radio-box-id-proc (Table 128), and displays the subworkspace (3006) of the new panel, a Pathway Query Panel. This panel is composed of: a) the title (3022) which specifies the type of search being made from that panel and names the bioReservoir (3023) that is the point of reference for the search; b) the set of radio-buttons (3024) of the radio-box labeled "direction", defining the two possible alternatives; c) the set of radio-buttons (3025) of the radio-box labeled "function", which defines the possible alternatives available for function, which in this case we only show a small but important subset that is of high relevance to scientists. d) the set of radio-buttons (3026) of the radio-box labeled "motifs", which defines the possible alternatives

available for subclasses of protein-motif. Many other possible sets of radio-buttons, such as those of the radio-box labeled "location", which defines the possible alternatives available for subcellular compartments, are not shown.

- The user may now use that Pathway Query Panel to select any combination of one radio-button per radio-box. Selecting the "Upstream" option (3007) calls the upstream-query-lists-callback which in turn calls the upstream-function-query-lists-proc. Selecting the "Downstream " option calls the downstream-query-lists-callback (Table 141) and then the downstream-function-query-lists-proc (Table 142), which do the following: a) it first calls repeatedly the create-br-query-list-proc (Table 143) to create a set of two lists per each predefined type of function, one to contain names of bioReservoirs and the other the names of bioEntities; b) then, it loops over all bioReservoirs that are upstream (or downstream) and scans for their connections to certain classes of bioReactants, which define their function, and enters their name in the corresponding one of those bioReservoirs lists, for example, if the bioPool of a bioReservoir is connected to both a kinase.r and a transcription-factor.r, then the name of such bioReservoir will be added to each of the two corresponding lists, so that a bioReservoir may be named in more than one function list, but is not duplicated within one single list; and c) for each pair of one of those bioReservoirs lists and its corresponding bioEntities list, it calls the fill-br.et-list-proc (Table 144) which scans all the bioReservoirs in the fist list for the value of their "ref-bioentity" attribute, which if available is added to the second list. All those lists, to be used by the procedures described below, are associated by a relation to the panel from which they are created, are valid only for the bioReservoir giving the related-item of that panel, and are deleted when the panel is deleted.
- The user can now select any combination of one radio-button per radio-box, such as receptors (3008) and ß-strand-motif (3009), and then select the "Query" button (3010) to complete the processing and display the results of the query. The processing from the "direction" button is independent and previous to the processing from the "Query" button, and if the user selects the "Query" button before the "direction" button, then processing cannot ocurre, because the lists do not exist, and an error message is displayed to the user to that effect. Selecting the "Query" button calls the query-br-related-callback (Table 145), which first obtains the values of the selected radio-buttons of the "direction" (3024) and "function" (3025) radio-boxes, and then calls the get-br-function-list-proc (Table 146) with those values among its arguments, which uses those values to obtain and return the corresponding set of two lists from the bioReservoirs lists and bioEntities lists created and filled in the previous processing, as discussed above. Such bioReservoirs list is going to be used to directly pass its values to the bioReservoirs scroll-area for final display, while such bioEntities list is going to be used as the basis for the complex further processing that follows.
- The callback scans now the Panel to find whether there are structure related radio-boxes, such as the "domains" radio-box in this example, in which case it calls the query-br-structure-lists-proc (Table 147) with those previous values and the bioEntities list among its arguments, which does a complex processing as following: a) it finds the label of such structure related radio-box(es) and calls find-specific-array-proc (Table 133) to find the value of the selected radio-button for such box(es), such as ß-strand-motif (3309), and its matching array(s) in the Query-Arrays-Bin (as described above in the Structure Related Queries section); b) if there is only one structure related radio-box, like in FIG.30, then it calls merge-br-function-structure-proc (Table 148), or if there are two structure related radio-boxes it calls merge-br-function-structure-structure-proc (Table 149), passing among its arguments the values of all selected radio-buttons, as well as the corresponding bioEntities list filled earlier, and the array(s) now found. These procedures create a new list, the bioEntities scroll-text-list, and then, taking into consideration the type and value of the arguments: a) they select a text specific for each situation, to be displayed as a title for the results, b) they call the br-fill-etl-list-proc (Table 150) or the the br-fill-double-etl-list-proc (Table 151), which enter the common elements of both the bioEntities list and the array(s) in the scroll-text-list, and c) they return the text and the scroll-text-list to the previous procedure from which they where invoked, which in turn returns them to the callback.
- The callback passes such text to a free-text structure, and finally calls the create-double-query-output-panel-proc (Table 152) with the bioReservoirs list mentioned in paragraph f), the bioEntities scroll-text-list, and their two corresponding free-texts among its arguments, which creates an output panel by cloning the Master-Molecular-Query-Output-Panel and stores it in the Temporary-Panel-Bin, then transfers the two free-texts to it, creates two scroll-areas to display the elements of each of those lists in a scrollable and interactive format, and calls the complete-br-panel-proc and complete-br-panel-proc (Table 152), which configure those scroll-areas and transfers them to the subworkspace, which is them displayed (3011).
- The user can now use the Pathway Query Output Panel (3011) so created to scroll through the lists bioReservoirs and bioEntities that are the output of the query. Selecting and unselecting any option of such scroll-areas allows to interactively display or hide subworkspace of the named structure by calling the query-message-selection-proc or the query-message-unselection-proc (Table 136). Each of the titles (3012) are

directly related to the scroll-area below them and vary with the results of each part of the search, and they may omit naming those groups of buttons where not selection was made, or indicate that some search or part of the search was empty. The scroll-area (3013) on the top shows the bioReservoirs that meet the direction and function requirements, which are those named by the options listed, and selecting any of its options (3014) displays the subworkspace (3015) of the named bioReservoir. The scroll-area (3017) on the bottom shows the bioEntities that meet the combination of direction, function and structure requirements, as selected, which are those named by the options listed, and selecting any option (3018) displays the subworkspace (3019) of the named bioEntity. Those structures displayed can be used to navigate in any way previously described, including up and down the pathways from bioReservoirs (3015) to bioProcesses (3016) and so on, from bioReservoirs or bioReactants or bioProducts to their bioEntities as shown in FIG.30, or up and down the workspace hierarchies (3020, 3021), by combining the variety of tools and methods previously described. As before, all the Pathway Query Selection Panels, Pathway Query Panels, and Pathway Query Output Panels are dynamically created and transient, and when the search is completed they are no longer needed and they are deleted, as well as their associated lists, by selecting the "Delete" button (3309), which calls the delete-query-panel-callback.

- More complex queries can be performed with an additional Master-BR-Molecular-Query-Panel with four additional options, to include queries previously described in combination with an additional constraint criteria, the location of the bioReservoirs selected by the previous criteria within the different cell compartments. Selection of any of those new options displays the corresponding Query Panel, and in this new set of Panels, the "Downstream" and "Upstream" radio buttons are associated instead with the downstream-function-location-callback (Table 153) and upstream-function-location-callback, respectively, which call the downstream-location-query-lists-proc (Table 154) or its upstream equivalent. Those procedures in turn call some of the procedures previously described and also call the find-compartment-proc (Table 155). The user may additionally select any combination of one radio-button each from the radio-boxes offered in each particular panel, and then selecting the "Query" button now calls the query-br-related-location-callback (Table 156), which in turn, depending on the boxes available on the particular Panel, calls one of an alternative set of procedures, which in turn call other procedures, as specified. The reader is further referred to the listings in Tables 157 through 169 for the details of such complex processing that is difficult to describe, and which uses a similar approach and methods as those previous described to create a new set of lists and to merge lists with bioReservoirs meeting the different criteria.
- The underlaying shell loads the desired application modules on memory for run time operations, but it is able to merge and remove modules while the system is running. Since the inference engine's reasoning and search space is limited to those modules that are loaded, it is necessary to create a mechanism to allow for searches in modules that are not yet loaded on memory. This is accomplished by maintaining in the repository module an object, called the query-arrays-bin, which holds in its subworkspace a set of item-arrays for intelligent dynamic merging and removal of the modules required at particular times, particularly when searching through the bioEntity-module and In order to allow for searches in a modularized application. Those contained-in-module arrays are built following similar methods as those described here for bioModels. The difference is that a new set of relation is defined, of the series -contained-in-module, and an initialization procedure that loops over all the workspaces of each loaded module, finding out the value of their assigned-module, and then looping over all the bioModel Libraries upon each workspace, and over all the bioModels upon each bioModel Library, and the continuing with the methods described in the previous section

f) **Interactive Lists and Pathways**

- In addition to the various query methods previously described, this invention provides several other tools and methods to enable the user to extract the knowledge build entered into the system by the modeler, in addition to the new knowledge created by applying the methods of this invention to analyze and integrate the knowledge provided by the modeler.
- One of those capabilities is provided by means of the "list copies" menu option (625) defined for the class complex-bioEntity, which upon selection calls the et-list-copies-callback (Table 170), which creates a scroll-text-list and then calls the list-local-bioentities-copies-proc (Table 171), which in turn scans all the bioEntities of the same major subclass as the one bioEntity from which the request originated to find if the master-bioentity of any of then matches the name of such bioEntity, in which case the value of the "id" attribute of each of those copies is added to the list. If there is any element in that list, then the callback creates an Output Panel by cloning the Master-Local-Bioentity-Copy-Panel, creates a specific title and a scroll-area, which are then added to the Panel and configured, and finally the Panel is displayed. This Panel looks like the Query Output Panel (3011) discussed above, with a different title. Selecting and unselecting any option of such scroll-area call et-copies-message-selection-proc or the et-copies-message-unselection-proc (Table 172) which

interactively display or hide the subworkspace of the structures referred to by their "id", rather than by name. This also provides an example of how, in general, instances of bioEntities as well as bioReservoirs, bioProcesses and bioModels, or any other structure, can be referred to by the value of any attribute, rather than by their names, which have to be unique and sometimes inconvenient, but they can be avoided. Referring to attributes, defined for such or any other purpose, also allows to refer to groups of instances within a class, without having to refer to each individual member of the group.

- Another novel teaching of this invention is the capability of creating lists of downstream and upstream bioReservoirs and bioProcesses. By selecting the icon of any of those lists, the user can display on a workspace the elements of that list, which are pointers to all the bioReservoirs or bioProcesses that meet the criteria set for each specific list, and each pointer further allows to either display the table of attributes of such object, or displays the object itself upon its workspace, as described in more detail later (FIG.34). Navigation can be now initiated from those objects as well as any other tasks described in this invention. This capability is accessible through the "show lists" options from the menus of bioReservoirs (3023) and bioProcesses (3103), which appear in the menus only when they are named and their subworkspaces are activated, as well as from the list-tracers in their subworkspaces, as described below.
- As illustrated in FIGs. 30 and 31, chaining-tracers (Table 30) are iconic programmed objects connected (3030) to a bioPool, or to a bioEngine (3105), where they are used to control the dynamic creation and display at run-time of a set of such lists of either bioReservoirs or bioProcesses that are either upstream or downstream of the bioReservoir upon which subworkspace such list-tracer is located. The "show" option (Table 31) is defined for the class list-tracer and is inherited by all its subclasses, including BR-chaining-tracer and BP-chaining-tracer. Selecting any such list-tracer by the user implies "show", which starts the list-tracer-handler-proc (Table 31), which configures the buttons and depending on the subclass of list-tracer, calls either the BR-chaining-proc (Table 173, also started upon selection of the "show lists" option of the bioReservoir), or the BP-chaining-proc (Table 174, also started upon selection of the "show lists" option of the bioProcess). The major actions performed by both of these procedures comprise: a) if the list-tracer already has a subworkspace, then it is displayed with its contents, otherwise a subworkspace for the BR-chaining-tracer or BP-chaining-tracer, respectively, is created by cloning the subworkspace of the prebuilt master structures BR-Chaining-Lists or BP-Chaining-Lists, respectively, which in both cases contain two sets of four predefined lists each, one set (3107) is meant to be used in any user mode, while the second set (3108) is to be used in simulation mode and contain elements from the first set that have been activated as a result of their inclusion in a simulation model (to be described in below), and then such lists are displayed; and b) those lists are populated having as point of reference the bioReservoir or bioProcess from which they were requested, by using for each list of the set the following criteria: the elements of DBPL and SDBPL are downstream bioProcesses, the elements of UBPL and SUBPL are upstream bioProcesses, the elements of DBRL and SDBRL are downstream bioReservoirs, and the elements of UBRL and UDBRL are upstream bioReservoirs.
- A further novel teaching of this invention is the capability to dynamically generate displays of interactive pathways, where these pathways comprise a chain of downstream, or upstream, bioReservoirs or bioProcesses or both, orderly arranged in a graphic format with connections drawn between them to indicate those that are adjacent. By visually following those connections, the user can follow the pathways that indicate a dependency relationship between those bioReservoirs or bioProcesses or both. The capability to display interactive downstream pathways is accessible from several different types of structures: a) from an individual bioProcess, in which case such bioProcess is the point of reference; b) from a Simulation Panel (to be discussed in conjunction with FIG.33), in which case a bioReservoir is the point of reference; and c) from an Experiment Panel (to be discussed in conjunction with FIG.35), in which case one or more bioReservoirs are the points of reference.
- As illustrated in FIG.31, now we will refer to accessing such capability from an individual bioProcess, or from icons in its subworkspace that have such specific functions, to display either upstream or downstream pathways, and in either case the bioProcess is the point of reference. Up-path-tracer and down-path-tracer are iconic objects (Table 28) located upon the subworkspace of a bioProcess (3110 and 3117), where they are used to control the dynamic creation at run-time of a graphic display of such interactive upstream or downstream pathway, respectively, from the bioProcess from which it was requested. The "show" option starts the path-tracer-handler-proc (Table 29), which depending on the subclass of path-tracer calls either the draw-up-pathway-proc (Table 175, also started from the "up pathway" option (3109 from the bioProcess) or the draw-down-pathway-proc (similar to the one above but downwards, also started from the "down pathway" option (3116) of the bioProcess. The major actions performed by each of these procedures comprise: a) it creates a subworkspace here called a Upstream Pathway Display for the up-path-tracer (3110) by cloning the subworkspace of Master-Up-Pathway, which contains a set of prebuilt auxiliary structures. The text of

the title (3111) is changed to refer to such bioProcess. The x-pos and y-pos (3112) are two integer-parameters Table 3) used to keep track of the most distant bioProcess added to the Display and which: i) are used by the program to compute the position of the next bioProcess to be added, after which their values are updated, ii) are used by the program to be compared to defined threshold values that, when reached as the display dynamically grows, trigger a reduction in the size of the display, and iii) may be used also by the user to read the size of the Display, which may in occasion get very large, by observing their value as displayed in digital form. The set of resize buttons (3113) allow the user to resize the Display at will, offering the options of full, three-quarter, half, and one-quarter scale.

b) it creates a copy of such bioProcess, by cloning it and deleting its workspace, and then transfers the copy to the Pathway Display at a specified position (3114), and displays the Pathway Display on the window from which it was invoked; and

c) it calls the create-local-up-bp-proc (Table 176) with such original bioProcess and its copy among its arguments, which in turn for each bioReservoir that is an-up-bioReservoir-of the original bioProcess finds each up bioProcess that is an-up-bioProcess-of such bioReservoir and then for each such up bioProcess: i) if a copy of such up bioProcess already exists upon the Display then it proceeds to iv, otherwise it creates a copy of such up (or down) bioProcess, by cloning it and deleting its workspace, ii) it computes a new value for x-pos and y-pos, iii) it transfers the copy to the Pathway Display at the new specified position (3615), iv) it creates a connection between random locations at the icon of such new (or existing) copy and the icon of the copy that is an argument of this invocation of the procedure, and v) it calls itself (Table 176), but now with such up bioProcess and its copy among its arguments. Each of these procedures keeps looping and calling itself until all upstream bioProcesses have been processed, and their copies added to the Display and connected. The an-up-bioReservoir-of and a-down-bioProcess-of, as well as the a-down-bioReservoir-of and an-up-bioProcess-of, are inverse relations, as previously defined (Table 75), which were established during initialization (Table 100) between appropriate pairs of a bioReservoir and a bioProcess each. A similar set of procedures is used to display the downstream pathways from the down-path-tracer.

- FIG.31 illustrates short examples of Upstream Pathway Display (3110) and Downstream Pathway Display (3118). The connections are directional, with the directions of the connections above the bioProcesses indicating the direction of material flows, that is from the one that is upstream to the one that is downstream in relation to each other. The directions of the connections below the bioProcesses always show the right to left direction as a result of the next copy being one that already existed in the display, such as those connections that revert to 3119 from other posterior copies. Those pathways are interactive, and selecting the "details" option (Table 120) of any of the icons, or changing to Navigation Mode and selecting the icons (3119, 3121), displays the subworkspaces (3120, 3122) of the originals of those copies, which allow further access to any of the navigation (3123) and all other capabilities associated with their iconic components, as described throughout the description of this invention, and allowing further access to possibly distant parts of the application.
- Pathways with only bioReservoirs (not shown), or pathways with alternating bioReservoirs and bioProcesses (not shown) can be created with similar methods, using only minor modifications. In this invention the pathways with bioProcesses are preferred because the provide the most information about the interactions within the space constrains, while the connected bioReservoirs can be accessed by just clicking on two icons. Mixed pathways may become very large.

g) **Navigation Panel**

- An interface and methods are provided to constraint all the capabilities provided in regard to navigation and display of pathways, to limit the scope of the search to an space selected by the user based on the different levels of compartmentalization. This feature can be accessed through: a) a Navigation Panel, which can be requested from each bioReservoir and is dynamically created with the bioReservoir as the reference point; or b) an Experiment Panel. The Navigation Panel is to be used in General Mode, and is not available in Simulation Mode, since the methods do not activate the subworkspaces of the selected structures, as do those associated with the Simulation Mode, described below.
- As shown in FIG.32, selecting the "navig-panel" option (3203) from a bioReservoir (3201) calls the create-direction-panel-callback (Table 177) which displays a Selection Panel (3204). Selecting now the Upstream radio button calls the select-direction-callback (Table 178), which in this case further calls the create-upstream-navig-panel-callback (similar to Table 179), which calls the navig-create-upstream-lists-proc (similar to Table 180), which creates two lists with all of its upstream bioReservoirs and bioProcesses, and the complete-navig-panel-callback (Table 186), which creates a Navigation Panel (3205) containing a scroll-area (3202) already configured to list all the possible levels of bioModels, the compartments within the Virtual Model, in which such bioReservoir or all of its upstream bioProcesses are contained. The user now has to

select the desired scope of the search space by selecting one of those bioModels. By selecting for example in this case the cytosol of the Go phase of a fibroblast (3202) and then selecting the LIST button (3207) calls the list-navigation-UBP-callback (similar to Table 181), which creates new lists and calls the navig-read-up-scroll-proc (similar to Table 182) to populate the new lists, displayed on the Panel (3208), with the members of the previous upstream lists that are contained in such bioModel (and all its encapsulated compartments). Those lists display the number of elements it contains, alerting the user about the potential size of the resulting pathway, and if too large or too small the operation can be reset and a different bioModel, lower or higher in the hierarchy, can be selected. Selecting any of those lists (3208) displays (3209) its members, and selecting any of those members (3202) allows direct access to the named bioObject (3210) on its original bioModel (3211), and displaying its details (3212) allows to further interactively navigate through the upstream or downstream pathways of such particular bioObject.

- The user can now select the PATHWAY button (3213), which calls the navig-draw-UBP-callback (similar to Table 183), which calls the navig-create-local-UBP-callback (similar to Table 184) to create a navig-path-tracer, by cloning a master structure which subworkspace is partially built, which is transferred to the Panel (3214) and its subworkspace is displayed and further configured by calling the navig-create-local-UBP-callback (similar to Table 184), which actions are similar to those previously discussed in reference to FIG.31, resulting in the display of the selected pathway (3215). By selecting any of those bioProcesses (3216) and displaying its details (3217) allows to further interactively navigate through the upstream or downstream pathways of such particular bioProcess. Selecting the RESET button calls the reset-navigation-callback (Table 187) which destroys the lists and the navig-path-tracer upon the Panel, enabling the selection of a different bioModel and repeat the process, and selecting the DELETE button deletes or recycles the Panel.

4. **Simulation Mode: Simulating the Virtual Models**

a) **Initialization**

- The Simulation Mode can be restricted to any desired set of individual users, by packaging the application accordingly. The domain-menu comprise the heads: "Pathway Libraries" and " Panels", and allows also changing between user modes. Before a simulation can be run, the application must be appropriately initialized for such purpose. This is accomplished by accessing the "Panels" menu and selecting the "Initialization Panel" option, which displays the Initialization Panel.
- Initialization is necessary only if the application has been started or reset, or if additions or changes have been made in the part of the application to be simulated since the last Initialization. The "Add Simul" button is to be selected only after the "Navig Init" was been successfully concluded. If changes to the database have been made by the modeler since the "Navig Init" was last concluded, or if the application has been restarted, then the "**Simul Init**" button should be selected, since it comprises the methods of the other two. The common purpose of both buttons is to scan the application for the availability of certain values of the bioReservoirs that define the initial conditions required by the simulation, and if those values have not been provided by the modeler, to infer those values from other values provided by the modeler, and if none of the alternative source of values are available, either default values will be used, including symbolic approximations, or, depending on the attribute which value is missing, to send messages to the user to provide any of the alternative values and also to provide messages to the to be used any time that the user request a predefined query that involves that kind of knowledge.
- Selecting the "Add Simul" button calls the add-simul-callback, which in turn calls the simul-init-proc (Table 188), while selecting the "Simul Init" button calls the simul-init-callback (Table 188), which differs in that it first calls the navig-init-proc (Table 99), before calling the simul-init-proc (Table 188), which in turn does the following:
  a) it calls the send-no-input-messages-proc (Table 188) which scans the loaded application for every bioReservoir that has a name but that has no inputs, either in the form of bioProcesses connected to its biopool-p-posts, or as input model-blocks connected to its bioPool, and it sends a message to the user for each of those bioReservoirs found with no inputs, as a warning for the user to take appropriated measures if those bioReservoirs are to be involved in a simulation. The simulation proceeds also without those values, since inputs can also be provided through the Simulation Panel as user inputs, or otherwise the quantity of such bioPool will initially be whatever value the modeler has set or the program will set for the basal-quantity parameter used, or its default value, and it may or may not be exhausted during the simulation run;
  b) it creates a set of four lists, which are transferred to the Initialization Panel, and which are provided to inform the user about the source of the values for the Scaling-Density attribute of each bioPool, by adding the name of each bioReservoir to one of those lists, which elements that meet the following criteria: i) the list labeled UNList contains all bioReservoirs which initial-value of the Scaling-Density was provided by the

modeler; ii) the list labeled ANList contains all bioReservoirs which initial-value of the Scaling-Density was set by the program to equal the value, provided by the modeler, of the if-scaling-amount of that bioReservoir; iii) the list labeled RNList contains all bioReservoirs which initial-value of the Scaling-Density was set by the program to equal the value of the if-scaling-amount of a bioReservoir which name was entered by the modeler in the "if-scaling-bioReservoir" attribute of the original bioReservoir; iv) the list labeled DNList contains all bioReservoirs which initial-value of the Scaling-Density was was set by the program to equal the default value of 100.0. To do so, for each bioReservoir it scans the value of its Scaling-Density and if it has a non-default value it enters the name of such bioReservoir to the first of those lists, otherwise it calls the set-scaling-density-proc (Table 188) which sorts and adds the bioReservoir to one of the other three lists, following the criteria mentioned above, after setting the value of the Scaling-Density of its bioPool accordingly.

- The set-scaling-density-proc (Table 190) discussed above is the method used in the current implementation of this invention in conjunction with the scaled reasoning used as default. An alternative method for setting the values of the scaling-density attribute, to be used when more quantitative information is available for the system to be modeled, and in which case the modeler may prefer the mixed absolute/scaled type of reasoning, in which case an alternative procedure, the long-set-scaling-density-proc (Table 189)should be called. This alternative method is much more complex but also more specific, and involves the known values of the kinetic-parameters of the participating bioReactants for each process. A mixed approach can also be developed where if those kinetic-parameters are known then the reasoning of this procedure is applied, and if they are not available, then the previous more generic procedure is called.
- The "**Icons Init**" button calls the init-icons-callback (Table 118), which function is not essential for the functioning of this system, but which provides a very useful cosmetic function, by realigning the tracers in the subworkspaces of bioReservoirs and bioProcesses and making their connections invisible. Selection of this button is not necessary every time the application is started, and it can be selected any time thereafter, when problems are detected, in any mode other than Simulation Mode (where those subworkspaces may be deactivated). The same effect is achieved for individual bioReservoirs or bioProcesses by slightly moving and repositioning their bioPool or bioEngine, respectively, which triggers their respective icon-movement rules, as defined in Table 81, which also apply to repositioning the bioRole-posts any time a bioReactant or bioProduct is moved.
- Selecting the "**Init All**" button calls all-init-callback (Table 119), which in turn calls the all the initialization procedures mentioned above, and selecting the "**Hide**" button calls the hide-workspace-callback, which hides this Panel.
- With the alternative way of simulation where the original bioObjects are used, when selecting Simulation Mode from another mode starts the version of the change-mode-proc shown in Table 7, which causes the subworkspaces of all bioReservoirs and all bioProcesses to be deactivated, and during the processing involved in activating a submodel for simulation, the subworkspaces of only the bioReservoirs and bioProcesses selected to participate in such simulation will be activated; while when leaving Simulation Mode to another mode causes the subworkspaces of all bioReservoirs and all bioProcesses to be activated again. The version of the change-mode-proc used with the other simulation alternative, where copies of only those bioObjects to be used are created, does not contain the lines revering to activation or activation of subworkspaces.

b) **Simulation Panel and Procedures**

- Because the applications build with this system can become very large, it is important to have control over which parts of the Virtual Model are included in a given simulation. To control performance, two implementations are provided with this system, each having advantages over the other depending on the hardware facilities available:
  - In the first implementation, the constraining of the simulation space is achieved by using activatable subworkspaces for composite bioObjects, such as bioReservoirs, bioProcesses, and certain time-compartments, and by having all those structures initialized to a deactivated state. When the subworkspace of an object is deactivated, any structure upon that subworkspace is not seen by the inference engine or the simulator. Only the subworkspaces of the desired structures at any given time are activated, and only their contents are therefore participating in a given simulation and associated inferences. Methods associated with buttons on the entry-panels used for simulations and experiments, activate the subworkspaces of bioObjects connected along the desired pathways and contained within the selected bioModel or compartment, and this activation is necessary for propagating values along the pathways. Inactivation of those subworkspaces occur upon resetting the selection within those panels. This mechanism is also very utile in activating or deactivating entire branches of a simulation model at run-time. The efficiency of large scale simulation applications is further improved by having the initiation of the simulation of different subcomponents of the high-level bioModel be driven by events, when appropriate, or by time intervals. This implementation allows only a single user to run

simulations concurrently, since the attribute values of the bioObjects change during the simulation, as well as the activation state of such subworkspaces. For multiple users, multiple processes of the program have to be started.

- A second implementation is based in dynamically cloning those bioReservoirs and bioProcesses, or certain time-compartments, required by the simulation, and to use those copies to hold the changing attribute values, rather than the originals. In this implementation, the subworkspaces of bioReservoirs and bioProcesses are not activatable, remaining always activated. On the other hand, the subworkspaces of certain time-compartments are still activatable, are deactivated when created, and their activation is controlled during the simulation, operating as in the first implementation. This implementation allows multiple users to run simulations concurrently, each on a different X-window, but increases the burden on the system by increasing the required RAM to hold the additional transient copies and to run more than one simulation concurrently. Since the system of the current invention is modularized, and the modules can be dynamically up-loaded and down-loaded, part of this problem can be solved by up-loading the module(s) that contain the originals, creating the copies for the various simulations on the top module, and removing the modules that are no longer required from the memory, before the simulation is started.

- A simulation can be initiated from any named and connected bioReservoir in the Virtual Model, from the bioPool of such bioReservoirs, or from the bioReactants and bioProducts connected to such bioPools. As shown in FIG.33, selecting the "simul panel" option (3303) from the desired bioReservoir (3301), which is only available when in Simulation Mode, calls the create-input-type-proc (Table 190) which creates one of two types of entry-panel by cloning one of the prebuilt master structures and displays the subworkspace (3305), called a Simulation Selection Panel, depending on whether the subclass of bioPool involved represents soluble or bound bioEntities (Concentrations or Density inputs, respectively). Either Panel (3304) prompts the user to select the type of input, and the choices offered are combinations of absolute or scaled values with one-time or periodic inputs. Selection of one of the radio-buttons calls the select-input-panel-callback (Table 191), which: a) according to the user's selection it calls one of a set of specific procedures, including such called in this example, the create-periodic-relative-input-panel-proc (Table 193), which in turn creates and partially configures one entry-panel of the specified type by cloning one of the several prebuilt masters; b) it creates two lists and then calls the fill-br-downstream-lists-proc (Table 194) to search the application for all bioReservoirs that are downstream of such reference bioReservoir, as defined by the relations established during the initial initialization as described earlier, and fills the DBR list with those bioReservoirs, and then do the same for the downstream bioProcesses to be added to the DBP list, and uses those values to configure the scroll-area, as discussed above in relation to FIG.32; and c) it calls complete-input-panel-proc (Table 192) to complete that panel and display its subworkspace (3305), called here a Simulation Panel. Each of the entry-panels is related to the bioReservoir, or the bioReservoir referred by the bioReactant, from which the request was originated, that is, the related-item attribute of the entry-panel newly created is a pointer to such bioReservoir. The difference between the one-time or periodic inputs types of Simulation Panel is that the latter has two additional entry-boxes to allow the user to enter the time interval for subsequent entries (3309) and the number of subsequent entries (3308) at such intervals of the amount indicated by the entry value (3306) and starting at the indicated time (3307); b) the difference between the absolute or scaled inputs types of Simulation Panel is that the entry value in the former is given in some units standardized throughout the application while the latter is a dimensionless value; and c) the prebuilt panels contain the overall layout with the common components, while other structures specific for each bioReservoir are added dynamically at run time.

- The Simulation Panel can now be used to interactively set-up and start simulations comprising a desired bioModel by selecting one from the scroll-area (3312), as described above in relation to FIG.32, to allow selection of an adequate size of the model to be simulated. The parameters and optional capabilities used for the simulation at run-time are configured by using an structure upon the Simulation Panel specific for that purpose, the model-definition. Its table of attributes (3319) allows the configuration of any desired attributes. The "items-belonging-to-this-model" (3320) is set by the program to name the bioModel selected by the user, and the variables and parameters that form part of the simulation model are all those encapsulated in any of the bioReservoirs and bioProcesses in the pathway contained in the workspace named by such attribute. This is important when dealing with very large systems, allowing to focus the computer resources on the subsystems of interest. Further attributes are used to set the time interval between simulation cycles, the type of integration algorithm desired for the state variables (3321); the configuration for any external simulator that can also be used to receive or provide values of the variables in this model; the name of any main procedure that is invoked once every simulation cycle, which in turn may start or call any other procedure, and which in this case calls the model-simulation-proc (Table 204); a choice to send all the values to external databases or simulators at the beginning of each cycle; an indicator of whether the simulation is not-running, running or paused, or whether is nothing to simulate or is a simulation error; and the configuration of the simulator clock to run either

synchronously or as fast as possible, which is more appropriate for simulating models which actions take place over long time periods).

- This Panel permits the user to enter the input value (3306) to be added to the Accumulation of such bioReservoir and the time (3307) of such input. Help buttons in each section of the Simulation Panel provides instructions for the user in how to proceed, such as the one (3310) which subworkspace is shown (3311). Following those instructions, selecting the LIST button (3313) calls the list-simul-callback (Table 195), which finds the previous lists and then calls the read-scroll-proc (Table 196) to search for the members of those lists that are contained in the bioModel selected by the user, as previously discussed, and fill the lists labeled DBRL (3314) with downstream bioReservoirs, the SDBPL list (3315) with downstream bioProcesses, and the the UBRL list with bioReservoirs that provide the inputs for such bioProcesses but which are not included in the first list.
- As shown in FIG.34, selecting one of the bioReservoir lists (3314) displays a workspace (3401) with pointers to each of the elements of that list, which allows to examine the bioReservoir that are to be included in the simulation. Selecting any of those pointers (3402) allows the user to either display the table of attributes (not shown) of that bioReservoir or to directly display the bioReservoir and its subworkspace (3419) of its bioReservoirs Bin. In a similar way, selecting the SDBPL list (3814) displays the pointers (3405) to the bioProcesses that are to be included in the simulation, which upon selection (3406) directly displays the bioProcess and its subworkspace (3425). The user can examine those lists to decide whether the selection made is appropriate before proceeding. If not satisfy the user can reset the panel, and transfer to the panel additional or different submodels, and press the "List" button again.
- When a Simulation Panel is created, the "Activate" and "Start" buttons are disabled to prevent the user of taking such steps prematurely. When the processing associated with the LIST button is completed, the ACTIVATE button (3315) is enabled, and when selected it calls its associated procedure, which is one of three listed (Table 197) and specific for each type of Simulation Panel. Such procedures check that a value has been entered by the user and then call activate-simulation-proc (Table 198), which in turn: a) looks on the Panel for the three lists created above and calls the activate-model-proc (Table 199) with those lists as arguments. As discussed in sections above, upon entering the Simulation Mode the subworkspace of all bioReservoirs and bioProcesses are deactivated, so that their variables become inaccessible to the simulator. The function of this procedure is now to scan the DBRL, DBPL, and UBRL lists and activate the subworkspaces of each of their elements, which are the ones that will be now included in the simulation, after which returns back to the previous procedure, which scans for the Basal-Density attribute of the now accessible bioPool of each of the bioReservoirs on those DBRL and UBRL lists and if its value has been modified by the user then it adds the name of the bioReservoir to the User-BC list of the set described below, and if its value has a default value then it calls the basal-density-proc (Table 200) for such bioReservoir.
- A simulation starts with the baseline model, after being initialized by setting the initial values of the initial amounts of the bioPools (concentrations, densities, scaled-amounts, or other quantities, depending on how the bioPools involved where defined) to be equal to the values of the corresponding basal amounts. In order to activate the system, a disturbance may be introduced by entering one or more input values through the entry-panel of the desired bioReservoir, or through the desired experiment-selections of the experiment-panel By entering different values for those variables, or by changing the values of the constant parameters, what-if analysis can be performed on the system. Montecarlo simulations can be performed using a montecarlo-model-block as an input to each of the bioPools desired as test inputs, as explained in other section, and the simulation can then be started from either the entry-panel of a desired bioReservoir, or through the desired experiment-selections of the experiment-panel, without entering entry-values. The history of values for each of the variables of interest can be stored in the form of a matrix-like structure that is an array of arrays of values for each variable or parameter. The values of those arrays are transient values that can be used for statistical and sensitivity analysis within this system, or can be archived to a text file external to the system of this invention, or the values can be transferred to an external statistical package for further analysis. In addition, if the values of those arrays are desired in a permanent form within this system, this is accomplished by using the procedure provided to make the initial values of an array equal to the list of its current values, separated by commas.
  - The basal-density-proc (Table 200) has an important inference role since the value of the Basal-Density (or the Basal-Concentration) is used when the simulation is started as the initial value for the state variable of the bioPool, the Accumulation. The task of this procedure is to obtain a value for that attribute, if the user has not specifically entered one, from other data and information contained in other attributes of that bioReservoir or its bioPool. It also uses a set of lists contained in the structure labeled BA (3317) upon the Simulation Panel to sort and list all the bioReservoirs to be included in the simulation, to indicate the source of those values used for the Basal-Density. The user can examine the results of that inference after the processing associated with the

ACTIVATE button is completed, when the button becomes disabled and its appearance changes. Selecting the BA button (3317) displays its subworkspace with a set of lists filled by the program following the criteria discussed below. Those list can now be examined, by selecting their icons as discussed elsewhere, to find out which of the bioReservoirs got the value from each of the alternative more or less precise sources.

- Various alternative versions of this procedure are made available, which are disabled and the one preferred is enabled by the modeler to better match the particular application. Alternatively, each of the alternative versions could be made specific for particular subclasses or groups of bioReservoirs. This procedure initially scans the normal-basal-density value of the bioReservoir and if that value is different from the default value of 9.99e-99 then the Basal-Density is set equal to its value, and the name of the bioReservoir is added to the NBD-BC list . If the subclass of bioReservoir is sol-mol-reservoir then the Basal-Concentration would be set equal to the value of the normal-basal-concentration if different from the default value. In the alternative implementation shown here, if the subclass of bioReservoir is sol-mol-reservoir this procedure scans the normal-basal-concentration value of the bioReservoir, if that value is different from the default value, 9.99e-99, then the Basal-Density is set equal to its value multiplied by the Avogadro's number, which states that each mol has 6.023e23 molecules, and the name of the bioReservoir is added to the NBC-BC list. The conversion is made when modeling complex mixtures of bound and soluble molecules. For example, for many biochemical systems where the molecules form large localized active complexes molecules are not homogeneously distributed, and therefore it is more meaningful to integrate in a bioProcess the quantities of both types of bioPools representing bound and soluble molecules using the number of molecules rather that their concentrations. If the normal-basal-density (or normal-basal-concentration) has the default value, then this procedure scans for the Scaling-Density attribute of the bioPool and if it has the default value then the bioReservoir is added to the "NO BC" list. Otherwise, this procedure scans for the Scaled-Basal-Amount of the bioPool, and if its value is not the default value, then the Basal-Density is set equal to the value resulting from multiplying the Scaling-Density by the Scaled-Basal-Amount values, and the name of the bioReservoir is added to the SBA-BC list. Otherwise the Basal-Density is set equal to the value resulting from multiplying the Scaling-Density by a factor associated according to the symbolic value of the "abundance" attribute of the bioReservoir, which value is set by the modeler or otherwise the default value is used, and the name of the bioReservoir is added to the PAB-BC list. The inference engine may proceeds the search for other sources of values, such as in an alternative method (not shown), where this procedure sets the value of the Basal-Density equal to a value proportional to the physiol-max-density (or physiol-max-concentration) of the bioReservoir if this value is different from default. If the user has not entered any of those values, then the program sets the value of the Basal-Density equal to the average of any known thresholding parameters of any bioReactant connected as an output of this bioReservoir, such as the Km, Ks, Ki or 1/(2*Kp). This alternative method emulates the following reasoning: the value of Basal-Density represents physiological conditions and it is assumed that physiological concentrations approach the mid-range functional concentrations in the most relevant bioEngine in which the bioReservoir participates, as represented by constants such as the Km for a substrate, the Ki for an inhibitor or the Ks for a ligand or complex-subunit. If those values are not known, a message is displayed requesting the user to assumed some values, based on comparisons with similar systems. Under normal in vitro assay conditions, an enzyme is generally present in limiting or catalytic amounts, in a range between 10e-3 nM and 10e2 nM, while a substrate is generally in the range of 10e3 nM to 10e7 nM. This value can be adjusted at a later time as more is learned about the system. If none of the alternative means are provided the system's default values are set.
- Once the participants in the model have been activated and configured, the input values have been entered, and the model-definition has been configured, the simulation is started by selecting the START button (3322), which depending on the type of Simulation Panel will call one of the six procedures specific for each of the one-time Panels (Table 201) or for each of the periodic Panels (Table 202), which read the input values entered by the user and perform different types of processing depending on the types of variables involved. The concentration-entry-panel is used to input the absolute concentration value entered by the user in the input-value edit-box, in units such as molar, which is added as a concentration or is converted and added as a density (depending of the alternative options available in this implementation). In a similar way the density-entry-panel is used to input an absolute density value, such as units/compartment, while the relative-entry-panel is used to input the scaled dimensionless value. Those input values are used the values of the corresponding attributes in each type of bioPool, such as the density-entry or the scaled entry at the time of entry value entered by the user, where these values are then integrated by the formula of the Accumulation of such bioPool at the next simulation cycle, and the values of such attributes are reset to 0.0 after one simulation period has elapsed since the time the input values were set. The set of periodic type of Panel follow initially the same type of processing, but the setting of the values of the density-entry or the scaled entry attributes of the bioPool occurs then repeatedly as many times as entered by the user in the entry frequency edit-box at times intervals as entered by the user in the time-interval edit-box, beginning at the simulation time entered by the user

in the input-time edit-box. Each of those procedures then call start-simulation-proc (Table 203), which requests the simulator to run the model as previously defined in the model-definition.

- During the simulation run the model-simulation-proc (Table 204) is called once every simulation cycle, which sets the values of the attributes current-simulation-time and current-simulation-time-increment of the entry-panel. The user can select the "T" button (3325) to display its subworkspace, which contains a digital display of those values as well as the values of other performance parameters. This procedure also calls the compute-graph-transform-proc (Table 205) described below.
- By selecting the PAUSE button (3323) or the RESUME button (3324), which call their associated procedures (Table 206), the user can pause and resume the simulation at will. By selecting the RESET button (3326) the reset-simulation-callback (Table 207) resets all the buttons on the Panel and calls the reset-proc (Table 207), which: a) first it scans the DBRL, UBRL, and DBPL lists and resets each of the bioReservoirs and bioProducts listed, including their status attributes, their icons and deactivating their subworkspaces; and b) then delete those lists. Note than when a Simulation Panel is created, most of its buttons are disabled, and they are enabled as the previous step has been selected by the user and completed by the program, at which point the button of that previous step is disabled. Disabled and disabled buttons have different appearances, and this implementation both guides the user about which steps are available at each point in time, including RESET which is enabled at all times, and furthermore, prevents the user from taking steps ahead of time by mistake. The DELETE button (3327) is disabled when the LISTS button is selected and it is not enabled until the reset is completed, preventing the user from deleting the panel before all the participants in the simulation that may have been modified are reset.
- The procedures that are modified for the alternative implementation of the simulation, when copies are used instead of the original bioProcesses and bioReservoirs so that the values of the originals are not modified, are listed in Tables 211 through 221).
- **Digital and Graphic Displays of Dynamic Values**

- This system's graphic interface allows simulations to be followed in different ways, including:
- creating pathways on a workspace where dummy copies of the bioProcesses involved, or full copies of both the bioReservoirs and bioProcesses involved (depending on the alternative method of simulation used) of all the participating), are located in sequence and the corresponding connections between them are drawn. This structures allow to further interactively navigate through the system as usual, including back to the originals from which the copies were made.
- animation by color changes of the activated components ;
- display of the current values of desired variables in digital form;
- dynamically created graphs of a set of time series of the values of desired variables; and
- dynamic charts of the values of a variable versus another variable;
- Selecting the details option of any of the bioReservoirs or bioProcesses that take part in the simulation allows to follow up the quantitative simulation by providing access to the different structures in their subworkspaces Only the icons upon subworkspaces that are activated are active. Therefore, when in Simulation Mode the capabilities now to be described, as well as the navigation capabilities, are restricted to only those structures that are selected participants for that simulation. The current values of the variables of each bioReservoir or bioProcess are by default displayed in digital form (3403 and 3404). To dynamically display the time course of the values those variables for the time frame selected during a simulation graph-tracers are used, which subworkspaces with a graph and associated structures are not stored, but are rather created or deleted when the user selects one of such tracers. These methods are described now.
  - A graph-tracer is an object (Table 32) connected either to a bioPool (3421 and 3428) or a bioEngine (342n), used to control the dynamic creation and display at run-time of a graph which plots the values over time of key variables or parameters of the bioObject where they are contained, scaled or absolute versions for bioReservoirs. Selecting the "show-plot" option calls the graph-tracer-handler-proc (Table 208), which depending on the class of the graph-tracer from which it was invoked calls either the scaled-BR-graph-proc (Table 208), the absolute-BR-graph-proc, or the scaled-BP-graph-proc, which first create a subworkspace for that tracer with all its contents by cloning the subworkspace of the corresponding prebuilt master structure, and then configure the labels of the graphs (3423 and 3428) and the ref-bioreservoir or ref-bioprocess attribute of each of the graph-tranf-vars upon such new subworkspace to refer specifically to that bioReservoir (3419) or bioProcess (3425), respectively.
  - The design of graphs and associated structures deal with plotting multiple variable which values differ by orders of magnitude on graph structures that do not provide multiple axis scales. In the current implementation, the system automatically adjusts the scale to the current values of the time-series to be plotted, using a system of

intermediary variables to dynamically transform those disparate values into others that fit a common scale. This set of new variables are instances of subclasses of the class transf-factor-var (Table 15), a float-variable. A representative example of the attribute table of a default instance of one of those subclasses, a velocity-transf-factor, is shown (Table 209). Among the attributes of transf-factor-var are: "transform", which value is given by an instance of the class graph-transf-var (Table 14); and "ref-variable", which makes reference to the particular attribute of a bioObject that provides the input value. The Label and the Transform value are displayed. The role of the graph-transf-var is to hold a transformed value of the current value of the parameter or variable referred to by its "ref-variable". The transformed value is suitable for plotting within the scale of the related graph. The role of the transf-factor-var is to hold the value of the factor necessary to obtain the desired transformation, as inferred by the compute-graph-transforms-proc (Table 205) only when a simulation-model is running. Each subclass of the class transf-factor-var corresponds to one of the bioObject's attributes to be transformed, which matches the value defined for the "label" attribute of each subclass. The transf-factor-var provides by itself the order of magnitude of the variable it transforms.

- Selecting the "hide-s.graph" option (Table 210) of the bioReservoir, which appears only when the graph upon the subworkspace of the scaled-BR-graph-tracer exists, or deselecting the graph-tracer, or selecting the delete option directly from the subworkspace of the tracer deletes that subworkspace with its contents. The same applies for the other classes of graph tracers.

- **Scaled Computation Approach for the Variables of bioReservoirs and bioProcesses**

- The formulas of the parameters and variables that dynamically characterize a bioPool, depend on other numeric variables or parameters that are either attributes of the bioPool or its bioReservoir, or that are attributes of the bioReactants and bioProducts that represent such bioReservoir in different bioProcesses. Table 83 lists a representative set of formulas, as examples of generic simulation formulas comprised in this invention to simulate a set of scaled-valued variables and Table 84 lists a set of generic simulation formulas specific for the Contribution of different subclasses of bioReactants that incorporate the values of scaled-valued variables or parameters. Table 86 lists a set of generic simulation formulas that are alternatives to the formulas in the set shown in Table 84, and which can be used in this invention to incorporate the values of absolute-valued variables or parameters into the Contribution of each bioReactant. These formulas for the Contribution do scale those absolute values using either a linear or sigmoid approach, depending the role in a bioProcess represented by each class of bioReactant. Therefore, since the output of the Contribution is dimensionless, whether the inputs are scaled (as in the case of using the formulas in Table 84) or absolute (as in the case of using the formulas in Table 86), it is now possible to integrate in a single bioProcess a mixture of bioReservoirs, where some of those bioReservoirs use only the scaled-valued variables while other bioReservoirs, called of mixed-type, use a mix of the scaled-valued variables, such as the Scaled-Amount, and absolute-valued variables, such as the Density. Those two values are interconvertible by using the value of the Scaling-Density, as computed by the formula for the Density in Table 86, so the user may be able to display the output of a simulation also as converted back into Density units in this case. The Contribution can be used in conjunction with either absolute-valued or scaled-valued variables, and therefore allows to integrate parts of the simulation where the absolute values of parameters and initial conditions are known, where other parts of the simulation where those values are unknown an have to be approximated, which is best done when using dimensionless values.

- The Quantity of the bioPool is the only quantitative output value propagated from any component of a bioReservoir to any component of a bioProcess. That output quantity value of the bioPool is used to compute either: a) the value of the Velocity of each bioEngine connected to each bioReactant connected to such bioPool, which is the method normally used when using the absolute computation approach, to be discussed below; and b) the value of the Contribution of each bioReactant connected to such bioPool, which is an intermediary variable which in turn is used to compute the value of the Velocity of the bioEngine to which that bioReactant is connected, in combination with the Contributions of all bioReactants connected to such bioEngine, this being the preferred method used when the scaled reasoning mode of simulation is preferred by the modeler or user. The formulas for the variables that give the value for the Contribution are specific for different classes of bioReactants, as more specifically defined in Tables 83 and 84. The formulas for the classes that represent reactants that bind to specific sites other reactants, such as substrate.r, ligand.r, inhibitor.r or ion.r, are designed as to provide a sigmoid scaling around their characteristic kinetic parameters, which implicitly provides a smoothed thresholding of the Quantity of the bioPool for that particular bioProcess, as represented by the bioReactant. When using the scaled-valued set of variables, the Scaled-Amount is used in conjunction with the corresponding scaled kinetic parameter, as the examples given in Table 86 show. The formulas for the Contribution for other classes of bioReactant, such as receptor.r or enzyme.r, are designed as to provide a linear scaling from none to their maximum observed value, represented as 100 if such absolute value is unknown. In

contrast with the absolute mode of computation, where there may be great variability in the formulas needed to provide the value for the Velocity, depending on the class of bioEngine involved, in the scaled mode of computation the variability is introduced, to a much lesser extent in the simulation formulas that provide the value for the Contribution, while only one very generic simulation formula is required for the Velocity of all classes of bioEngines. As listed in Table 83, the very generic simulation formula for the Velocity included in the current implementation of this invention integrates various coefficients and a rate-constant to give the modeler or user the flexibility of by changing the values of those attributes of the bioEngine to modify the behavior of the system in different ways, without the need to write specific formulas for each situation, while the default values of those attributes do not affect the system.

- The Velocity of each bioEngine in turn is used to compute: a) the Production-Rate any bioProduct connected to such bioEngine; and b) the Consumption-Rate any bioReactant connected to such bioEngine, which may be 0.0 if such bioReactant represents a reactant that is not consumed in such process, or may have a value that may be withdrawn from a bioPool as an output (like any standard Consumption-Rate), but then retained for a period of time and then returned back to the bioPool as an input (like any standard Production-Rate will do), as specified by methods specific for the variables of such particular pair of bioReactant and bioPool, or for a group of such pair which are identifiable by some specific common attribute, or which are defined as separate subclasses of their respective classes.
- Several bioProcesses may contribute through their outputs to the Input-Rate of that bioPool, which values are then integrated when computing the value of the Accumulation of that bioPool. Users can also add (or remove) an absolute or scaled amount during a simulation to the selected bioPool, by having specific procedures setting the values of specific attributes of the bioPool, as discussed above, which are then integrated into the Accumulation by its simulation formula.
- The Basal-Quantity (which may be either a Basal-Concentration, a Basal-Density, or a Scaled-Basal-Amount) represents the initial conditions, and provides the initial-value for the integration of Accumulation, which is a state variable that integrates all other dependent variables. When using the scaled-valued set of variables, the value of the Scaled-Amount is the value of the Accumulation constrained to a range within 0.0 and 1.0. The values of the Concentration or Density are constrained to not be less that 0.0.
- In this invention, the value of variables or parameters that are attributes of bioObjects upon the subworkspace of bioReservoirs and bioProcesses, can be simulated only if the subworkspace of the bioReservoir or bioProcess upon which it is located is activated. When the user selects simulation mode, the subworkspaces of all bioReservoirs and bioProcesses are deactivated, and only those that are to participate in a simulation run are activated. This is a mechanism to improve performance of what may become an extremely large system of parameters and variables. In the current implementation of this invention, each of the values is computed once each simulation cycle using the equations defined by simulation formulas.
- The implementation of the intermediary variable, the Contribution, is a novel and very important teaching of this invention, particularly in applications when the knowledge of the quantitative parameters and initial conditions of the system are incomplete, or when a more abstract and generic system is desired, such as when providing a Shell to be used in different and unpredicted ways. This implementation is relevant in several different ways. It allows to treat each of the bioReactant of a bioProcess as a generic self-contained unit, which allows to model also participants in a bioProcess which are known to interact but which exact roles in such process are unknown, where the velocity is generically computed by simply multiplying the contributions of any participant that the modeler may want to model. These generic units become even more generic when used in combination with the set of scaled-valued variables, which allow the user to enter values based on expert knowledge, or quickly perform what-if analysis. Another important result of using the Contribution is that, as discussed above, allow to perform quantitative simulations that integrate into the Velocity of any bioProcess values originated from bioReservoirs with both types of absolute-valued and scaled-valued variables. This means that those parts of the complex model where the absolute values of their kinetic parameters and initial conditions are known can be integrated with other parts of the complex model where relative values are used because the absolute values may be unknown.
- The abstraction provided by the Contribution also facilitates the use of bioProcesses as black-boxes, which are also be implemented in the current system of this invention without using the Contribution. Black-boxes are used to represent any type of participant that is known to cause something to change somewhere down the line, even if it is known that other intermediaries are involved, but where the details or identity or even the existence of those intermediaries is unknown. The bioReactants in a black-box represent the entities (of bioPool(s)) that cause some effect on some other distant entities or bioProducts (the input to other bioPool(s)), and like in other generic bioProcesses, either the kinetic-coefficient (scaled or absolute) that modifies each bioReactant, or the

rate-constant that modifies the overall process, representing the proportionality factor between the quantity of cause(s) and the quantity of effects, can be used.

- **Absolute Computation Approach for the Variables of bioReservoirs and bioProcesses**

- More standard ways of quantitative simulations using the known absolute values of the system parameters and initial conditions, can be performed by using only the set of absolute-valued variables or parameters, and by using a different set of generic simulation formulas, such as those shown in Table 85. From those variables, the one that offers more variability in the formulas needed for each class of bioEngine is the Velocity, which is in contrast with the scaled mode of computation where only one very generic simulation formula is required for the Velocity of all classes of bioEngines. It is also possible to write very complex simulation formulas to encompass a variety of different situations, such as that in Table 82 to be described below. The value of the Velocity of a bioEngine in this approach depends directly on the Quantities of the bioPools connected to each bioReactant connected to such bioEngine, which is the method normally used when the formulas that give the value of the Velocity are those that represent the equations that a biochemist will normally use to compute the velocity of a reaction, based on the quantities of the reactants and on some known kinetic parameters, such as the Michaelis constant (Km), inhibition constant (Ki) or the equilibrium dissociation constant, and which the preferred method used when the absolute reasoning mode of simulation is preferred by the modeler or user. The simulations formulas for the Velocity of absolute-valued enzymes, such as that listed in Table 85 (which is equivalent to the Briggs and Haldane approach where V = n* kp*[E]*[S]/([S] + Km * X), where X = 1 if no inhibitor.r is connected to the bioEngine, or X = (1+ [I]/Ki) if an inhibitor.r is also connected to the bioEngine), assume the instantaneous or initial velocity approach, since increases or decreases in the Concentration of the substrate are updated at small time intervals in this invention.

- Following are some examples of how the common knowledge of chemists is implemented in the current invention.
- For receptor-bioEngines or enzyme-bioEngines with multiple equal binding sites, the velocities can represent reactions of two major types: a) the generic simulation formula for non-cooperative-binding is equivalent to V = n*Kp*[E]*[S]/(Ks+[S]), and represents the situation where binding at one site has no effect on the intrinsic dissociation constants of the vacant sites, yielding hyperbolic velocity curves, and being impossible to tell from the kinetics whether there is 1 pmole with n identical sites or n pmoles of 1 site; and b) the generic simulation formula for cooperative-binding represents the simple sequential interaction model of an allosteric enzyme with 2 cooperative substrate sites. It is modeled as two substrate.r connected to the same bioPool, but each of those bioReactants have different dissociation constants, and the simulation formula is equivalent to $V=n*kp*[E]*([S]/Ks1+[S]^2/Ks2^2)/(1+2*[S]/Ks1+[S]^2/Ks2^2)$. Here, the binding of one substrate molecule increases the affinities of the vacant sites for the next substrate molecules, yielding sigmoid velocity curves. Thus, the sigmoid response acts as a smooth "on-off" switch", and at intermediate specific velocities. this provides a much more sensitive control of the reaction rate by variations in the substrate Concentration. Allosteric enzymes generally result in sigmoid velocity curves due to cooperative binding, either positive or negative (binding one molecule facilitates binding of the next). The Hill Equation is used for an enzyme with n equivalent S binding sites, if the cooperativity is very marked, in which case the Km of the single substrate.r has to be set, not equal to the real Km but rather equal apparent Km to the n power. Under these conditions, dominated by [S]^n, the equation can be simplified to $V = kp*[E]*[S]^n/(Km'+[S]^n)$. For enzyme-bioEngines, the application in this invention focus on regulatory pathways, which are controlled by enzymes which active form is only present in catalytic amounts and a steady-state is assumed. For receptor-bioEngines, which include processes for other specific binding proteins, the generic formula for the velocity represents the system at equilibrium, and is equivalent to $v=[RL]=n[R]_t/(1+(Ks/[L]))$.
- Inhibition can be modeled in different ways, and the modeler has to decide what best represents each situation. Table 82 lists a very generic formula that allows the modeler to connect to the bioEngine one inhibitor-reactant that can be of any one of the four common types represented by the four defined subclasses of inhibitor-reactant. This formula is capable of first detecting which subclass of inhibitor-reactant is connected to the bioEngine, and then applying the equation with one of the alternative modifiers included which apply to that particular type of inhibitor. However, some of those types of inhibitors may be better represented in the system of this invention as competing with the substrate from a separate bioProcess, which allows for a more intuitive representation and understanding of such competition and a higher level of manipulation by the user, which are in fact some of the advantages offered by the system of this invention when compared with other more global modeling approaches. Competitive inhibition may be modeled within the corresponding bioEngine, with the generic formula such as that in Table 85. Noncompetitive inhibition is best modeled as a separate complex-formation-process between the enzyme and the inhibitor that competes for the enzyme by being connected to the same bioPool. Feedback inhibition should always be modeled by a separate bioProcess. Any further velocity equations can be derived in the same manner, by using well established equations in the field of

biochemical kinetics. and the reader is referred to standard references, such as Segel,I.H. ,1976, Biochemical Calculations 2nd Edition. John Wiley & Sons, New York.

c) **Experiment Panel and Procedures**

- An Experiment Panel is a facility that allows to perform many of the previously discussed, but with the additional feature: the capability of using multiple points of reference for the generation of lists, creation of pathways, or as the recipients of quantitative inputs for simulations. In contrast to the Simulation Panels, which are are dynamically created and deleted when requested from the bioReservoir that serves as point of reference, and , a Library of Experiment Panels are created as permanent generic structures with different designs, accessible through the "Panels" head of the domain-menus associated with the General Mode (2920) and with the Simulation Mode previously discussed. Each Experiment Panel is the subworkspace of an instance of entry-panel created by cloning the Master-Experiment-Setup-Panel. As shown in FIG.35, selecting one of those options (3502) from the domain-menu (3501) displays the subworkspace (3502) of the named entry-panel, which corresponds to one of those predesigned Experiment Panels. The new components of such Panel are round structures (3504) organized in columns and rows which are instances of **class experiment-selection** (Table 222). New designs are created by delening of adding any number of such auxiliary structures, used to select the combination of bioReservoirs that are to serve as the points of reference in such experiments, and also to allow the user to enter the input values for quantitative simulations of such experiments.
- Selecting one of the experiment-selections (3504) displays its table of attributes (3505) which allows to: a) enter a short name in its "label" attribute displayed on top of the icon, b) select a bioReservoir, as entered in its "ref-bioreservoir" slot. and c) enter whether "currently selected" is true or false. The values of those attributes are sufficient to perform any of the qualitative tasks associated with this panel. If a quantitative simulation is desired, any of the values for the other optional attributes may be entered, such as those for an absolute-valued input or an scaled-valued input, time of input, time interval for periodic entries, and entry frequency, which have the same use as those previously discussed in relation to FIG.33. The configurations of any or all those experiment-selections can be stored for later use, and the combinations of different experiment-selections to be included in a particular experiment may be quickly changed at will by changing the value of the "currently selected", since only those with such value set to true, which get also highlighted with a different color, are included in the processing when any of the tasks is requested. This allows the user to have a set of experiment designs for those types of experiments most frequently used. The BIORESERVOIRS button (3506) provides a graphic tool to facilitate the task of the user in selecting the appropriate name to enter in the "ref-bioreservoir" attribute of those experiment-selections, to be selected from among the large number of bioReservoirs in the Virtual Model. Selecting this button displays its subworkspace (3507) if one exists, or calls the BR-scroll-callback (Table 223) which creates one, containing an scroll-area listing in alphabetical order all the named bioReservoirs loaded on memory, and which in addition to providing the specific nomenclature used for desired bioReservoirs, it allows to directly access and verify the details of a selected bioReservoir.
- Upon entering the choices of "ref-bioreservoir" and setting the value true for the desired combination of experiment-selections, the user can proceed by selecting the PROCESSS button, which calls the process-exper-callback (Table 224), which creates a set of lists and then scans the Panel to find all experiment-selections which Currently-selected is true and the Ref-bioreservoir has a value, which are inserted in the "Exper. Selections" list (3512), and for each such experiment-selection found, it calls the same fill-br-downstream-lists-proc (Table 194) called from the Simulation Panel to fill the other two lists, one (3513) listing all downstream bioReservoirs of all those bioReservoirs named by the experiment-selections in the previous list, and the other (3514) listing all downstream bioProcesses of all those bioReservoirs named by the experiment-selections in the first list: and then finds all the bioModels that include "all" of those bioReservoirs and bioProcesses listed in those lists, and creates an scroll-area displaying those bioModels, to allow the user to select the desired scope of inclusion, as described in reference to FIG.32. Selecting the CLEAR button (3529) calls clear-exper-callback (Table 225) which deletes those lists and the scroll-area, to allow making different selections and repeating the PROCESS. The other buttons on the Panel have similar or equal functions as those described for the Simulation Panel, and therefore here only the differences will be briefly highlighted. The reader is referred to the Tables 225 through 233 for a more detailed description of the processing involved. Selecting the LIST button (3515) calls the list-exper-callback (Table 226) which creates a new set of lists: SDBRL (3516), SDBPL (3517), and UBRL (3518) lists, and finds the members of the previous sets of lists that are contained in the selected bioModel.
- Selecting the PATHWAY button (3520) calls the draw-exper-pathway-callback (Table 227), which creates a navig-path-tracer (3531) together with its subworkspace by cloning the Master-Navig-Pathway and transfers it to the Panel, and then it scans the "Exper. Selections" list and for each bioReservoir listed calls the

create-local-exper-BP-proc (Table 228), while setting the value of the x-pos to a value close to the initial position when changing from one bioReservoir to the next, which performs a processing very similar to that described earlier for the create-local-UBP-proc (Table 176) for the upstream direction. Now, instead of once as before, that procedure is called as many times as bioReservoirs are in the "Exper. Selections" list. One example of the display of one of those multi-origin pathways is shown in FIG.36, which are more complex. The pathways start from multiple initial bioReservoirs, and the pathways from latter initial points are connected, whenever the possibility arises, to the pathways from former initial points already on the Experiment Pathway Display. Alternative implementations of any of the types of pathways is to include in the drawing of the pathways not only the bioProcesses, but also the bioReservoirs that are intermediaries between those process in the chaining process. The procedures for those alternative implementations (not shown) apply the same methods and need only minor modifications to create, transfer and connect the intermediary bioReservoirs.

- This Experiment Panel can be used either for the only purpose of generating those complex pathways, or they can be further used to perform quantitative simulations that apply to the variables and parameter of the components of those pathways. For that purpose, the remaining buttons and structures to be used in the subsequent steps have been described before, and the procedures called upon selection of the such buttons (Tables 229 through 233) are only minor modifications of those described for the corresponding buttons of the Simulation Panel, to incorporate the fact that now there is not just one bioReservoir but the various bioReservoirs in the "Exper. Selections" list to consider.

d) **Simulating Complex Models**

- To deal with complex systems where some parts of the system may be more precisely defined than others, the system of this invention integrates a variety of methods to allow modeling and simulations while accommodating the uneven types of knowledge available. The following describes how the heterogeneous forms of knowledge are integrated in the system of this invention:
- In some cases, sufficient information has allowed scientists to develop mathematical models to represent a particular behavior of a system, and some of those models may be suitable for adaptation to be included in the knowledge-base, giving the user the option to integrate them in their simulations. This type of knowledge is usually included in the form of the parameters characteristic of the system and the initial conditions once the underlaying architecture of the network is matched, or it can be implemented directly by modeling particular inputs or outputs as model-blocks.
- Intermediate levels of information about a sequence of events, or effects following a cause or combination of causes, allows the construction of semi-quantitative behavioral networks that are constrained by the data available, such as time intervals, context and compartments in which behaviors occurred, concentration levels at which different behaviors were observed, and so on.... This type of information is used to design the compartmentalized architecture of the network, and to provide semi-quantitative values to the scaled parameters and initial conditions.
- Heuristic and shallow knowledge can be integrated in the form of either rules or formulas for inferred variables, that connect a cause with a distant effect, without detailed description of the unknown underlaying mechanism.

- Depending on the amount and type of information and data available about the system to be modeled, and the nature and size of the bioModels to be developed, the system of this invention's may operate in either quantitative or semi-quantitative simulation mode. With well known metabolic pathways, the simulation may be run in quantitative mode, but since in most biological systems both qualitative and quantitative information is incomplete, it may be difficult to simulate with accuracy such biological systems. However, the semi-quantitative methods integrated in the system of this invention allows to gain insight about the degree and direction of change in the model variables, and their quantity levels at different points in time.

- Of major importance in simulating the behavior of biological systems is the need to model the different states in which complex biological entities, such as cells, can be found at different points in time, and also to model the events that cause the transitions from one state to another. There are several major types of states and transitions to be considered, depending of whether the biological entity to be considered is a biological system, organ, cell, cellular compartment, molecule or any other entity. We are providing here with just a few examples among the many considered in the currently preferred embodiment of this invention.

- The modeler is provided with several methods to fine tune any specific parts of the complex model, including in the ability to write formulas specific for the individual instances of variables that are attributes of the bioReservoirs or bioProducts that model any such specific part of the complex model. Within those formulas, temporal reasoning is further achieved in the system of this invention in several ways, such as by specifying times or time intervals, using quantitative time constraints such as the expressions: during the last 30 seconds, as of 5 minutes ago, or between 3 hours ago and 4 hours ago. Fuzzy time intervals and time constraints may be

specified if so desired to handle the uncertainty encountered in biochemical systems. The simulation can be synchronous or asynchronous, by setting different simulation time intervals for sets of variables of bioProcesses and bioReservoirs that represent different parts of the complex model.

- The simulation can be continuous or time-constrained, and either way may be differentially applied to different parts of the complex model. To accomplish that, the simulation formulas can be complemented with rules that monitor those simulated variables to either set other values, particularly those that are further integrated into an state variable, or by controlling parts of the model. For example, a bioProcess or a bioReservoir can be programmed to be active only for certain time intervals during a simulation run, by means of rules or procedures that control the activation and deactivation of its subworkspace. Those time intervals may be either directly specified or dependent on the values of other parameters or variables or any of their combinations. Rules may be defined to monitor the simulated values of specific variables, such as the Quantity of a given bioPool or the Velocity of a given bioEngine, to be compared with a specified threshold, that when reached causes that rule to trigger some actions, such as setting the values of other parameters or variables, or activating and deactivating the subworkspace of one or more bioProcesses or bioReservoirs, or starting one or more procedures. Such rules in turn may be: a) if-type rules with their "scan-interval" attributes set to a specified time interval that preferably matches the time-interval used for the simulated variables, which means that such rules are invoked once every such interval; or b) whenever-type rules that are invoked whenever the specified variable receives a value and which trigger specified actions when such value fulfills any specified constraints; or c) initially-type rules, that may for example be hidden in the subworkspace of a bioEngine or bioPool, and which will be invoked once every time the subworkspace of their superior bioProcess or bioReservoir is activated, and which may trigger any of the actions described above which may apply to any bioProcess or bioReservoir in the complex model.
- One of the several applications of this invention is that of a system for signal processing and signal integration. In such system, simulations can have three levels of control through:
- the interactions explicitly modeled in the architecture of the graphic design of this invention, namely the network of connected bioReservoirs and bioProcesses, which represent and determines the specific interactions of the different components of the complex model;
- the propagation through such network of the dynamic changes in the values of the variables of such components; and
- through the signals transmitted by the smoothed thresholding implicit in the design of the Contribution variables of the bioReactants which determine the output values of the Velocity of the bioEngines, and
- through the rates of consumption of the bioReactants and the rates of production of the bioProducts, that determine new changes in the values of the Concentrations, now amplified to a larger number of bioPools, which in turn initiate new waves of value propagations.
- The simulation starts after a perturbation, such as a Concentration-Entry, Density-Entry or Scaled-Entry, has been introduced and integrated in the overall system's equation: $d[E](t)/dt = [E](t) + entry + \Sigma inputs(t) - \mu^*[E](t) - \Sigma outputs(t)$ In this system, the bioReservoirs and bioProcesses placed in a biological compartment-layer are activated successively in time and space, as the activation signal is propagating forwards based on various chaining relationships between those structures established during the initialization process according to the downstream or upstream positions of each of those structures respect to the others.
- "When starting the knowledge base, the G2-s finds the initial values for all variables that receive values from simulation formulas, and then executes the main simulation procedure. For each simulation cycle, the G2-simulator evaluates in order: 1. simulated state variables and parameters , 2. dependent variables and parameters, and 3. the main simulation procedure. The G2-s only evaluates one, the most specific simulation formula for each variable"
- BioReservoirs which variables have scaled values can be mixed in a simulation with BioReservoirs which variables have quantitative values, since both types of outputs are integrated at the bioProcess level only after being transformed to a scaled value. However, the output values of the variables of BioReservoirs and of the thresholding parameters, such as Km, Kd, Ki or kp, have to be of the same type, either quantitative or scaled. The choice of scaled values between 0.0 and 100.0 for basal-Concentration, based on the knowledge of the domain expert, can also be entered manually within the table of attributes of the bioReservoir.
- Tabular functions of one argument allow to deal with situations when the algebraic relationship between two bioVariables is not known but experimental data is available, which allows to build a table that relates a set of values that represents the magnitude of a cause with the set of values that represents the magnitude of the corresponding mechanistic effects or cellular responses, and straight-line interpolation is optional. These are important tools to biologist who frequently measure complex cellular responses to external factors.

5. Alternative Implementations and Variations

a) An equivalent system can be build with a less graphic interface by substituting each of the components represented in the current implementation as iconic bioObjects upon of the subworkspace of bioReservoirs and bioProcesses, and representing them instead as attributes of the bioReservoirs and bioProcesses, respectively. Examples of this alternative implementation are shown in Tables _ and _.. The corresponding methods would need only minor modifications, such as for example, by substituting the current expression "the bioPool upon the subworkspace of bioReservoir X" with "the bioPool that is an attribute of bioReservoir X"; or substituting the current expression "the bioReactant-1 connected at port-1 of the bioEngine upon the subworkspace of bioProcess X" with "bioReactant-1 of bioReservoir X", where each of the specific bioReactants connected to each of the specific stubs of a specific type of bioEngine are now converted into specific attributes of the specific type of bioProcess, and where those attributes are not values but objects, and where those objects may have as attributes other objects. For example, the bioPosts now connected with each bioReactant and bioProduct can also be implemented as and attribute of such bioReactant or bioProduct, while the two sets of bioPosts connected to a bioPool can now become two sets of attributes objects of the bioReservoir, such as Input-1 or Output-3. By defining a specific attribute, such as "post" for these new classes of objects, the specific distant connections can now be established by giving the same unique value to the "post" attributes of one of the bioReactant attributes of a bioProcess and one of the Output attributes of a bioReservoir, or by giving the same unique value to the "post" attributes of one of the bioProduct attributes of a bioProcess and one of the Input attributes of a bioReservoir. The result of this alternative implementation is that the different levels of encapsulation currently provided through the workspace hierarchy would then be provided by an attribute hierarchy, and the current representation of those components by their icons would be then limited to their representation by their tables of attributes, which are also part of the current implementation. Many of the tasks associated with the menus of the icons of those components or other auxiliary icons can be associated in the alternative implementation as menu options of the superior bioReservoirs and bioProcesses, as many of those are already currently implemented. The advantage of using the approach described in the currently preferred embodiment of this invention is that the more graphic interface is more intuitive for the user, and easier to use. The disadvantage is the additional memory required to store the additional graphical structures.

b) Another alternative is to replace the complex knowledge-structures bioReservoir and bioProcess by the components encapsulated in their respective subworkspaces, after some modifications. One of the several possible alternatives is to bypass the bioReservoir and the bioProcess structures altogether, and to encode the pertinent information here defined within the bioReservoir's table of attributes in the bioPool's table of attributes, and to encode the pertinent information here defined within the bioProcess's table of attributes in the bioEngine's table of attributes, and to construct the iconic components now contained in the subworkspaces of the bioReservoirs and bioProcesses directly on the desired workspace. The advantage of using the approach described in the currently preferred embodiment of this invention is that the bioReservoir and bioProcess structures encapsulate the details and simplifies the schematics, and makes easy the cloning and transfer of the preconstructed structures from one location to another, making the task of developing new models or modifying existing time much faster and keeping complex schemanes simpler. The disadvantage is the additional memory required to store the additional graphical structures.

c) An alternative more compact implementation that would still allow developing multidimensional pathways, bypassing the need for bioReservoirs, is to directly connect all the bioReactant-post and bioproduct-post that in the current implementation are connected to the same bioPool, by giving them all the same name. In such implementation, the "ref-bioentity" attribute, as currently defined for the class bioReservoir could instead be defined for the classes bioReactant and bioProduct, allowing in that way access to the bioEntity corresponding to an instance of bioReactant or bioProduct that would provide the physical description of such bioReactant or bioProduct. This implementation would allow the user integrating the description of structure with function. In another related implementation, the variables and parameters, inputs and outputs, as well as other attributes, as currently implemented as attributes of the bioReservoir or its bioPool, or as separate icons within the bioReservoir, could be implemented as attributes of the currently corresponding bioEngine.

d) Other of the many alternatives include: storing the bioReservoir or equivalent structure within the subworkspace of the bioEntity; or storing the bioEntity within the subworkspace of the bioReservoir. However, note that in the current implementation, the bioEntities are stored independently of the bioReservoirs because in many cases, and particularly dealing with very large applications, it may not be necessary to store one bioEntity for each bioReservoir, and instead prototypic bioEntities can be used that are shared by many bioReservoirs. But even in such cases, another alternative is storing all the bioReservoirs that in the current implementation would refer to the same bioEntity in the subworkspace of such bioEntity.

e) The concepts and methods subject of this invention can be applied with each of the object-oriented expert system shells from various vendors, after those capabilities not provided by that particular shell are additionally encoded. An equivalent of those capabilities of the selected shell that are required to apply the concepts and methods of this invention may also be custom built from scratch or by combining a set of off-the-shelf components.

f) There are many sophisticated capabilities that can be further developed with this system, which a knowledgeable expert can derive from the teachings of this invention. Here, only a few important ones of which will be described in detail:

g) As an alternative, the methods here represented could in part be compiled using this Shell or other shells of the compiled type, including but not limited to RTworks and Activation Framework. The compilation may improve performance at runtime by eliminating graphical interpretation of the iconic components . However, compilation does not facilitate frequent modifications of the design and information contained in the iconic components , a feature which is of great importance when modeling biochemical systems, the knowledge of which is being continually updated and improved.

h) The same concepts described in this invention can be slightly modified by a person skilled in the art, to incorporate multiple inheritance capabilities, allowing the bioObjects to inherit attributes from two or more superior classes. Additional controls and code may be required in that case to resolve conflicts that may appear when more than one superior class have the same attributes but receiving values from different sources, or through different methods.

i) Many of the symbolic variables or parameters could be also represented as logical variables or parameters, with or without fuzzy-beliefs, by simply changing the language of the attribute name and its values. For instance, an attribute called *status* could have values: *available, unknown and unavailable*; or it could alternatively be called *is-available* and have values *true, unknown or false*, the unknown associated *with fuzzy-beliefs* , such as the range 0.2-0.8.

j) To further control the behavior of any component of a simulation, different types of filters could be applied to, for example, the Density (or Concentration) of bioPools or to the Velocity of bioEngines. These filters can be of any of the types of filters used in signal processing control systems, such as low-pass or ideal high-pass filters (such as $H(j\omega)=e^{-j\omega to}$ for $|\omega| > \omega_c$ else $H(j\omega)=0$), Gaussian filters ($H(\omega)=e^{-\alpha(\omega)^2}e^{-j\omega to}$ and others). Such filters could be executed in either formulas or procedures, that could be generic for a whole class of variables, or more specific for certain instances or group of instances. The specific filters are more useful if fine control is required.

k) The Density (or Concentration) of a bioPool is the result of multiple interactions and is represented as a differential equation in the present embodiment. Alternatively, it could be represented as a discrete-time linear system or a continuous-time linear system of equations, or whenever the Shell is able to handle vectors, as their equivalent matrix system. Difference or differential equations can be converted to state form, and the state variables of the system are them described as a state vector, the coefficients of the variables are described as a system matrix and the forcing terms are described as the forcing vector, following standard procedures

TABLE 1

I. Group of variableStructures
- Group of Parameters
- Group of Variables
- Class Lists
- Class Array

II. Group of visualObjects

A. Auxiliary Tools

Class Button

Class Graph-Tracer
- Class scaled-BR-graphTracer
- Class absolute-BR-graphTracer
- Class scaled-BP-graphTracer

Class Lists-Tracer
- Class BR-chaining-Tracer
- Class BP-chaining-Tracer
- Class Query-Tracer
- Class Basal-density-source-Tracer
- Class BR-scroll-Tracer

Class Path-Tracer
- Class Down-path-Tracer
- Class Up-path-Tracer
- Class Complex-path-Tracer

Class Action-Button
- Class go-to-sw-Button
- Class paletteButton
- Class go-to-ws-Button
- Class hide-ws-Button
- Class go-to-sup-obj-Button
- Class Change-Mode-Button
- Class Resize-Button

Class bioConnection
- Class P-connection
- Class R-connection
- Class Link
- Class Line
- Class Graph-Link
- Class Icon-Wire

Class bioPost
- Class bioPool-Post
  - Class bioPool-R-Post
  - Class bioPool-P-Post
- Class bioRole-Post
  - Class bioReactant-Post
  - Class bioProduct-Post

Other Tools
- Class Inference-block
- Class Model-block
- Class Model-box
- Class Experiment-Selection
- Class bioEntity-Notes
- Class bioReference

...

B. Class bioObject

Class bioView-Object

Class bioModel

Class bioProcess

Class bioReservoir

Class bioEntity

Class bioRole-Object
- Class bioReactant
- Class bioProduct

Class bioNode-Object

Class bioPool
- Class sol-mol-pool
- Class bound-mol-pool
  - Class membr-bound-mol-pool

Class bioEngine
(see subclasses in TABLE 62)

C. Group of Panels
- Selection-Panel
- Query-Panel
- Entry-Panel

...

D. Group of Panel.Controls
- Class Toggle-Buttom
- Class Radio-Buttom
- Class Check-Box
- Class Type-in-Box

E. Group of Displays
- Class Graph
- Class Scroll Area
- Class Message

...

IV. Group of Inference and Simulation Structures
- Class Relation
- Class Rule
- Class Tabular-Function
- Class Generic-Formula
- Class Generic-Simulation-Formula
- Class Specific-Formula
- Class Specific-Simulation-Formula
- Class Units-of-Measure-Declaration
- Class User-Menu-Choice
- Class Procedure

TABLE 02

Class name	outcome-par
Superior class	symbolic-parameter
Attributes specific to class	none
Inherited attributes	none
Default settings	initial value for symbolic-parameter: within-threshold

TABLE 03

Class name	int-par
Superior class	integer-parameter
Attributes specific to class	none
Default settings	options for parameter: do forward chain
Icon description	width 40; height 40;

Class name	generation-number-par
Superior class	int-par
Attributes specific to class	none
Default settings	initial value for integer-parameter: 1

Class name	x-pos
Superior class	int-par
Attributes specific to class	none
Default settings	none
Attribute displays	last-recorded-value at standard position
Icon description	width 50; height 18;

Class name	y-pos
Superior class	int-par
Attributes specific to class	none
Default settings	none
Attribute displays	last-recorded-value at standard position
Icon description	width 50; height 18;

TABLE 4

Class name	float-par
Superior class	float-parameter
Attributes specific to class	none
Default settings	options for parameter: do forward chain; initial value for float-parameter: 0.0

Class name	phenotype-par
Superior class	float-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: per-cent

Class name	cell-number-par
Superior class	float-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: cells

Class name	cell-radius-par
Superior class	float-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: microns; initial value for quantitative-parameter: 10

Class name	cell-mass-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0

Class name	compartment-surface-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for quantitative-parameter: 9.9e-99; data type for quantitative-parameter: sq-microns

Class name	compartment-volume-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for quantitative-parameter: 9.9e-99; data type for quantitative-parameter: liters

Class name	cell-volume-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for quantitative-parameter: 1.5e-11; data type for quantitative-parameter: liters

TABLE 5

Class name	temperature-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 37.0; data type for quantitative-parameter: degrees
Class name	temp-factor-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 1.0
Class name	pH-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 7.4
Class name	pH-factor-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 1.0
Class name	bias-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0
Class name	alpha-coeff-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 1.0
Class name	tau-coeff-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 1.0
Class name	rate-constant-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0
Class name	cell-progression-rate-constant-par
Superior class	rate-constant-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0

TABLE 6

Class name	physiol-amount-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 9.9e-99
Class name	physiol-conc-par
Superior class	physiol-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: molar
Class name	physiol-density-par
Superior class	physiol-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: upce
Class name	physiol-cell-conc-par
Superior class	physiol-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: ce-p-ml
Class name	physiol-cell-density-par
Superior class	physiol-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: ce-p-co
Class name	scaling-density-par
Superior class	float-par
Attributes specific to class	set-by-user is false
Default settings	initial value for quantitative-parameter: 99.999; data type for quantitative-parameter: upce

TABLE 7

Class name	basal-amount-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 9.9e-99
Class name	basal-conc-par
Superior class	basal-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: m
Class name	basal-quantity-par
Superior class	basal-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: units
Class name	basal-density-par
Superior class	basal-amount-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: upce
Class name	basal-cell-conc-par
Superior class	basal-amount-par
Inherited attributes	none
Default settings	data type for quantitative-parameter: cepl
Class name	basal-cell-density-par
Superior class	basal-amount-par
Inherited attributes	none
Default settings	data type for quantitative-parameter: cepco

TABLE 8

Class name	time-delay-par
Superior class	int-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: seconds; initial value for integer-parameter: 0
Class name	elapsed-simul-time-par
Superior class	int-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: seconds
Class name	activation-interval-par
Superior class	float-par
Attributes specific to class	none
Default settings	data type for quantitative-parameter: seconds
Class name	clock-interval-par
Superior class	activation-interval-par
Attributes specific to class	none
Class name	time-interval-par
Superior class	activation-interval-par
Attributes specific to class	none
Class name	deactivation-interval-par
Superior class	float-par
Attributes specific to class	none
Class name	timer-count-down-par
Superior class	int-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 86400.0 data type for quantitative-parameter: seconds
Class name	timer-count-up-par
Superior class	float-par
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0; data type for quantitative-parameter: seconds

TABLE 9

Class name	float-pvar
Superior class	float-parameter
Attributes specific to class	none
Inherited attributes	none
Default settings	options for parameter: do forward chain
Icon description	float-pvar-pattern (width 50; height 40)
Class name	entry-pvar
Superior class	float-pvar
Attributes specific to class	none
Inherited attributes	none
Default settings	history keeping spec: keep history with maximum number of data points = 4
Class name	scaled-entry-pvar
Superior class	entry-pvar
Attributes specific to class	none
Default settings	initial value for quantitative-parameter: 0.0
Class name	amount-entry-pvar
Superior class	entry-pvar
Attributes specific to class	none
Default settings	initial value for quantitative-parameter: 0.0
Class name	conc-entry-pvar
Superior class	amount-entry-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: m
Class name	density-entry-pvar
Superior class	amount-entry-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: upcc
Class name	quantity-entry-pvar
Superior class	amount-entry-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: units
Class name	cell-conc-entry-pvar
Superior class	amount-entry-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: cepl
Class name	cell-density-entry-pvar
Superior class	amount-entry-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: cepco

TABLE 10

Class name	amount-pvar
Superior class	float-pvar
Attributes specific to class	none
Default settings	initial value for float-parameter: 9.9e-99; history keeping spec: keep history with maximum age of data points = 10 minutes
Class name	accumulation-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	none
Class name	scaled-amount-pvar
Superior class	float-pvar
Attributes specific to class	none
Default settings	initial value for float-parameter: 9.9e-99; history keeping spec: keep history with maximum age of data points = 10 minutes
Class name	conc-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: m
Class name	density-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: upce
Class name	cell-conc-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: cepl
Class name	cell-density-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: cepco
Class name	quantity-pvar
Superior class	amount-pvar
Attributes specific to class	none
Default settings	data type for quantitative-parameter: units
Class name	contribution-pvar
Superior class	float-pvar
Attributes specific to class	none
Default settings	initial value for float-parameter: 9.9e-99; history keeping spec: keep history with maximum age of data points = 10 minutes

TABLE 42

Class name	bioentity
Superior class	bioobject
Attributes specific to class	id is ""; references is "";
Class restrictions	unless in administrator, or developer mode: menu choices exclude additionally: go-to-subworkspace, disable, describe, name, move; when in simulation, navigation, or general mode: menu choices for bioentity exclude additionally: transfer, clone, change-size, color, create-local, delete; non-menu choices for item exclude additionally: move-object, move-connection, click-to-edit
Menu option	not a final menu choice
Inherited attributes	label is "", with an index; description is ""
Icon description	bioentity-pattern (width 100; height 80; dark-color = dim-gray, light-color = light-gray, white-color = white, type-color = purple, status-color = medium-aquamarine)

TABLE 43

Class name	simple-bioentity
Superior class	bioentity
Attributes specific to class	position is ""; superior-bioentity, with an index;
Class restrictions	when in developer or modeler mode: menu choices for simple-bioentity include additionally: rotate-reflect
Menu option	not a final menu choice

Class name	protein-site
Superior class	simple-bioentity
Attributes specific to class	none
Menu option	not a final menu choice
Attribute displays	position at standard position; label offset by (-30, 35)
Stubs	a link located at bottom 14
Icon description	width 28; height 40;

Class name	ligand.site
Superior class	protein-site
Attributes specific to class	none
Icon description	ligand.site-pattern (width 36; height 26; lime-green)

Class name	tyr.site
Superior class	protein-site
Attributes specific to class	none
Icon description	tyr.site-pattern (width 28; height 40; green)

Class name	protein-modified-group
Superior class	simple-bioentity
Attributes specific to class	modifiers is ""
Menu option	not a final menu choice
Attribute displays	position at standard position; label offset by (-30, 35)
Stubs	a link located at bottom 15
Icon description	protein-modified-group-pattern (width 30; height 32)

Class name	gly.isoprenyl.linker
Superior class	protein-modified-group
Attributes specific to class	none
Stubs	a link located at right 20
Icon description	gly.isoprenyl.linker-pattern (width 66; height 36; violet)

Class name	tyr.p-group
Superior class	protein-modified-group
Attributes specific to class	none
Stubs	inherited
Icon description	tyr.p-group-pattern (width 30; height 32; green)

TABLE 44

Class name	complex-bioentity
Superior class	bioentity
Attributes specific to class	toggle-state has values show or hide (default is HIDE) warnings is ""; master-bioentity; with an index
Menu option	not a final menu choice

Class name	bioentity-component
Superior class	complex-bioentity
Attributes specific to class	superior-bioentity; position is ""
Class restrictions	when in developer or modeler mode: menu choices for bioentity-component include additionally: rotate-reflect
Menu option	not a final menu choice

Class name	membrane
Superior class	bioentity-component
Attributes specific to class	none
Menu option	not a final menu choice
Icon description	membrane-pattern (width 20; height 130;

Class name	plasma-membrane
Superior class	membrane
Attributes specific to class	none
Icon description	plasma-membrane-pattern

Class name	protein-component
Superior class	bioentity-component
Attributes specific to class	none
Menu option	not a final menu choice
Attribute displays	label offset by (-45, -45)

Class name	protein-domain
Superior class	protein-component
Attributes specific to class	none
Attribute displays	label offset by (-42, 16); position offset by (-47, -25)
Stubs	a link located at top 40; a link located at right 23; a link located at bottom 40; a link located at left 23
Icon description	protein-domain-pattern (width 80; height 46)

Class name	fibronectin-type-III-domain
Superior class	protein-domain
Attributes specific to class	none
Icon description	fibronectin-type-III-domain-pattern

Class name	dna-component
Superior class	bioentity.component
Attributes specific to class	size is 0 BASE-PAIRS
Menu option	not a final menu choice

Class name	dna-motif
Superior class	dna-component
Attributes specific to class	position is ""
Menu option	a final menu choice
Stubs	a link located at top 40; a link located at right 23; a link located at bottom 40; a link located at left 23
Icon description	dna-domain-pattern (width 80; height 46)

Class name	tata-box
Superior class	dna-motif
Attributes specific to class	none
Icon description	tata-box-pattern

TABLE 45

Class name	molecule
Superior class	complex-bioentity
Attributes specific to class	synonims is "": in-species is " ": in-tissues is " ": in-cells is " ": in-organelles is " "
Menu option	not a final menu choice
Stubs	a link located at top 50: a link located at right 40: a link located at bottom 50: a link located at left 40

Class name	protein
Superior class	molecule
Attributes specific to class	cas-number is 00000-00-0. with an index: number-amino-acids sequence is " ": mol-weight is 0 DALTONS. with an index: isoelecuric-point is 0.0:
Icon description	protein-pattern (width 100: height 80: dark-color = dim-gray. light-color = light-gray. white-color = white. type-color = yellow. status-color = medium-aquamarine:

Class name	enzyme
Superior class	protein
Attributes specific to class	cas-number is "EC 0-0-0-000". with an index: substrates-info is "": inhibitors-info is "": ligand-info is ""
Menu option	not a final menu choice
Icon description	enzyme-pattern (width 100: height 80: dark-color = dim-gray. light-color = light-gray. type-color = red. status-color = dark-gray:

Class name	p.tyr.kinase
Superior class	enzyme
Attributes specific to class	none
Icon description	p.tyr.kinase-pattern (type-color = coral)

a p.tyr.kinase

Names	none
Label	""
Description	""
Id	""
Toggle state	hide
References	""
Master bioentity	none
Synonims	""
In species	""
In tissues	""
In cells	""
In organelles	""
Cas number	"EC 0-0-0-000"
Mol weight	0 DALTONS
Isoelectric point	0.0
Sequence	""
Substrates info	""
Inhibitors info	""
Ligands info	""

TABLE 46

Class name	bioentity-notes
Superior class	biotool
Attributes specific to class	label is "", with an index: toggle-state has values show or hide (default is HIDE)
Class restrictions	unless in administrator or developer mode: selecting any bioentity-notes implies details
Attribute displays	label offset by (-42, 20)
Icon description	bioentity-notes-pattern (width 74; height 48; dark-color = dim-gray, light-color = light-gray, white-color = white;

Class name	image
Superior class	bioentity-notes
Attributes specific to class	label is "", with an index: references is "":
Icon description	function.notes-pattern

Class name	mol.sequence
Superior class	bioentity-notes
Attributes specific to class	label is "", with an index: lenght is "": description is "": references is "": id, with an index: action-proc-name is crb-go-to-subws: toggle-state has values show or hide (default is HIDE)
Icon description	mol.sequence-pattern

Class name	function.notes
Superior class	bioentity-notes
Attributes specific to class	label is "", with an index: references is "":
Icon description	function.notes-pattern

Class name	reference.notes
Superior class	bioentity-notes
Attributes specific to class	none
Icon description	reference.notes-pattern

Class name	dendrogram
Superior class	bioentity-notes
Attributes specific to class	label is "Dendrogram": references is "": comments is ""
Icon description	dendrogram-pattern

TABLE 47

Class name	bioreservoir
Superior class	bioview-object
Attributes specific to class	references is given by a references-array: compartment is "": ref-bioentity is none: master-bioreservoir is none: status has values available. selected. selected-for-input. selected-for-output. down-selected. up-selected. forw-activated. back-activated. downstream-activated. upstream-activated. or deactivated (default is AVAILABLE). decay-rate-factor is 2.0e-4: if-scaling-bioreservoir is none: if-scaling-amount is 100: scaled-basal-amount is 9.9e-9: physiol-abundance has values highest. abundant. steady-state. low-induced. or only-induced (default is ONLY-INDUCED)
Capabilities and restrictions	activatable-subworkspace
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: go-to-subworkspace. clone: attributes visible exclude additionally: status: toggle-state. master-bioreservoir: when in simulation. navigation or general mode: menu choices exclude additionally: move. name. color. delete. disable. change-size. create-subworkspace. transfer. describe. clean-clone: non-menu choices exclude additionally: move-object: when in general mode: menu choices exclude additionally: simul-panel. show-s.graph. hide-s.graph: attributes visible exclude additionally: decay-rate-factor. scaled-basal-amount: if-scaling-bioreservoir. if-scaling-amount. when in navigation mode: selection of any bioreservoir implies details
Menu option	not a final menu choice
Attribute displays	label offset by (-57. -30)
Icon description	bioreservoir-pattern (width 96: height 56: dark-color = dim-gray. light-color = light-gray. white-color = white. status-color = medium-aquamarine. flag-color = medium-aquamarine. type-color = pale-green.

Class name	bound-mol-reservoir
Superior class	bioreservoir
Attributes specific to class	normal-basal-density is 9.9e-99: physiol-max-density is 9.9e-99

Class name	macro-complex-reservoir
Superior class	bound-mol-reservoir
Attributes specific to class	none
Icon description	inherited

Class name	cell-receptor-reservoir
Superior class	bound-mol-reservoir
Attributes specific to class	none

Class name	membr-bound-reservoir
Superior class	bound-mol-reservoir
Attributes specific to class	none

TABLE 48

Class name	sol-mol-reservoir
Superior class	bioreservoir
Attributes specific to class	normal-basal-concentration is 9.9e-99: physiol-max-concentration is 9.9e-99
Class name	int-sol-mol-reservoir
Superior class	sol-mol-reservoir
Attributes specific to class	none
Class name	ext-sol-mol-reservoir
Superior class	sol-mol-reservoir
Attributes specific to class	none
Class name	genet-mol-reservoir
Superior class	bioreservoir
Attributes specific to class	normal-basal-density is 9.9e-99: physiol-max-density is 9.9e-99
Class name	exp-cell-reservoir
Superior class	bioreservoir
Attributes specific to class	normal-basal-concentration is 9.9e-99: physiol-max-concentration is 9.9e-99
Class name	cell-reservoir
Superior class	bioreservoir
Attributes specific to class	normal-basal-density is 9.9e-99: physiol-max-density is 9.9e-99
Icon description	cell-reservoir-pattern (width 96: height 56: type-color = coral)

a cell-reservoir

Names	none
Label	""
Description	""
References	""
Warnings	""
Toggle state	hide
Compartment	""
Ref bioentity	none
Master bioreservoir	none
Status	available
Decay rate factor	2.0e-4
If scaling bioreservoir	none
If scaling amount	100
Scaled basal amount	9.9e-9
Physiol abundance	only-induced
Normal-basal-density	9.9e-99 UPCE
Physiol-max-density	9.9e-99 UPCE

TABLE 49

Class name	biopool
Superior class	bionode-object
Attributes specific to class	basal-density is given by a basal-density-par: scaling-density is given by a scaling-density-par: density-entry is given by a density-entry-pvar: scaled-entry is given by a scaled-entry-pvar: input-rate is given by an input-rate-pvar: output-rate is given by an output-rate-pvar: accumulation is given by an accum-pvar: density is given by a density-pvar: scaled-amount is given by a scaled-amount-pvar:
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: describe. rotate-reflect. move. transfer when in simulation or explorer mode: menu choices exclude additionally: add-scaled-input. add-input. add-output. add-top-post. add-bottom-post: when in explorer mode: menu choices exclude additionally: simul-panel: attributes visible exclude additionally: basal-density. density-entry. density
Inherited attributes	label is "", with an index: description is ""
Attribute displays	input-rate. output-rate. accumulation-rate. scaled-amount offset by (105. 20)
Stubs	an input-box-connection inp located at top 8: a scaled-input-box-connection sip located at top 17: an output-box-connection oup located at bottom 15: a p-connection p01 located at top 37: a p-connection p02 located at top 50: a p-connection p03 located at top 62: a p-connection p04 located at top 75: a p-connection p05 located at top 87: a p-connection p06 located at top 100: a p-connection p07 located at top 112: a p-connection p08 located at top 125: a p-connection p09 located at top 137: a p-connection p10 located at top 150: a p-connection p11 located at top 162: a p-connection p12 located at top 175: a r-connection r01 located at bottom 37: a r-connection r02 located at bottom 50: a r-connection r03 located at bottom 62: a r-connection r04 located at bottom 75: a r-connection r05 located at bottom 87: a r-connection r06 located at bottom 100: a r-connection r07 located at bottom 112: a r-connection r08 located at bottom 125: a r-connection r09 located at bottom 137: a r-connection r10 located at bottom 150: a r-connection r11 located at bottom 162: a r-connection r12 located at bottom 175 a graph-link s-plot located at left 11: a graph-link a-plot located at left 31: a graph-link lists located at right 11: a graph-link query located at right 31:
Icon description	biopool-pattern (width 180: height 42: dark-color = dim-gray. light-color = light-gray. white-color = white. type-color = green-yellow

TABLE 50

Class name	bound-mol-pool
Superior class	biopool
Attributes specific to class	none
Attribute displays	inherited
Stubs	inherited
Icon description	biopool-pattern (width 180; height 42; dark-color = dim-gray, light-color = light-gray, white-color = white, status-color = forest-green, type-color = medium-goldenrod)
Class name	cell-receptor-pool
Superior class	bound-mol-pool
Attributes specific to class	none
Menu option	a final menu choice
Icon description	cell-receptor-pool-pattern
Class name	membr-bound-pool
Superior class	bound-mol-pool
Attributes specific to class	none
Icon description	membr-bound-pool-pattern
Class name	macro-complex-pool
Superior class	bound-mol-pool
Attributes specific to class	none
Icon description	cell-receptor-pool-pattern width 180; height 42; type-color = gold)
Class name	soluble-mol-pool
Superior class	biopool
Attributes specific to class	basal-concentration is given by a basal-conc-par; concentration is given by a conc-pvar
Attribute displays	basal-concentration, concentration offset by (80, 60); input-rate, output-rate, accumulation-rate, scaled-amount offset by (105, 20)
Class name	int-sol-mol-pool
Superior class	soluble-mol-pool
Attributes specific to class	none
Icon description	int-sol-mol-pool-pattern
Class name	ext-sol-mol-pool
Superior class	soluble-mol-pool
Attributes specific to class	consum-cell-div-factor is given by a consum-cell-div-factor-pvar
Icon description	ext-sol-mol-pool-pattern

an ext-sol-mol-pool

Names	none
Label	""
Description	""
Compartment	none
Basal-density	9.9e-99 UPCE
Scaling-density	99.999 UPCE
Density-entry	0.0 UPCE
Scaled-entry	0.0
Input-rate	9.9e-99
Output-rate	9.9e-99
Accumulation	9.9e-99
Density	9.9e-99 UPCE
Scaled-amount	9.9e-9
Basal-concentration	9.9e-99 M
Concentration	9.9e-99 M
Consum-cell-div-factor	1.0

TABLE 51

Class name	genet-mol-pool
Superior class	biopool
Attributes specific to class	basal-amount is given by a basal-quantity-par
Stubs	inherited
Icon description	genet-mol-pool-pattern
Class name	exp-cell-pool
Superior class	biopool
Attributes specific to class	basal-concentration is given by a basal-cell-conc-par: amount-entry is given by a cell-density-entry-pvar: density is given by a cell-density-pvar: concentration is given by a cell-conc-pvar
Icon description	exp-cell-pool-pattern
Class name	cell-pool
Superior class	biopool
Attributes specific to class	basal-density is given by a basal-cell-density-par: density -entry is given by a cell-density-entry-pvar: density is given by a cell-density-pvar
Icon description	cell-pool-pattern

TABLE 52

Class name	biopool-post
Superior class	biopost
Attributes specific to class	id is "": ref-bioprocess is none
Class restrictions	when in navigation or simulation mode: selecting any biopost implies show-bp: when in explorer mode: selecting any biopost implies activate-bp: when in developer or modeler mode: menu choices for biopost exclude additionally: name. move. rotate-reflect. change-size. color. delete. describe. create-subworkspace. show-bp
Icon description	biopool-post-pattern (width 17: height 17)
Class name	biopool-r-post
Superior class	biopool-post
Attributes specific to class	none
Icon description	biopool-r-post-pattern (width 17: height 17)
Class name	biopool-p-post
Superior class	biopool-post
Attributes specific to class	none

TABLE 53

Class name	model-box
Superior class	biotool
Attributes specific to class	toggle-state has values show or hide (default is HIDE)
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: move. name. change-size. color. describe. go-to-subworkspace: selecting any model-box implies show-sw
Menu option	not a final menu choice
Icon description	model-box-pattern
Class name	input-model-box
Superior class	model-box
Attributes specific to class	none
Stubs	a p-connection located at right 10
Icon description	input-model-box-pattern

TABLE 54

Class name	bioprocess
Superior class	bioview-object
Attributes specific to class	compartment is "": status has values available. selected. selected-for-input. down-selected. up-selected. forw-activated. back-activated. downstream-activated. upstream-activated. or deactivated (default is AVAILABLE): deactivation-hold-interval is given by a time-delay-par: master-bioprocess is none
Capabilities and restrictions	activatable-subworkspace
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: go-to-subworkspace. rotate-reflect. change-size. clone: attributes visible exclude additionally: toggle-state: master-bioprocess: when in general. navigation. or simulation mode: menu choices exclude additionally:transfer. move. name. color. delete. disable. describe. create-subworkspace. clone. labeled-clone. clean-clone: attributes visible exclude additionally: notes. label: when in general or navigation mode: attributes visible exclude additionally: activation-hold-interval. deactivation-hold-interval: when in navigation mode: selecting any bioprocess implies details when in simulation mode: menu choices exclude additionally: up-pathway. down-pathway
Menu option	not a final menu choice
Inherited attributes	references is "": warnings is "": toggle-state has values show or hide (default is HIDE). with an index: label is "". with an index: description is ""
Attribute displays	label offset by (-57. -30)
Stubs	none
Icon description	bioprocess-pattern (width 96: height 56: dark-color = dim-gray. light-color = light-gray. white-color = white. status-color = slate-blue. type-color = slate-blue. flag-color = medium-aquamarine

Class name	cell.bioprocess
Superior class	bioprocess
Attributes specific to class	none
Icon description	cell.bioprocess-pattern

a cell.bioprocess	
Names	none
Label	""
Description	""
References	""
Warnings	""
Toggle state	hide
Compartment	""
Status	available
Activation hold interval	0 seconds
Deactivation hold interval	0 seconds
Master bioprocess	none

TABLE 55

```
Class bioEngine
    Class Binding-bioEngine
        Class Receptor-bioEngine
            Class bioengine.R1.L1
            Class bioengine.R2.L1
            Class bioengine.R1.L1.AC1
            Class bioengine.R1.L1.AN1
            Class bioengine.R2.L1.AC2
            Class bioengine.R2.L2.AN1
            ...
        Class Complex-Formation-bioEngine
            Class bioengine.U3
            Class bioengine.U5
            Class bioengine.U7
            Class bioengine.KU3
            Class bioengine.M1C3
            Class bioengine.C3
            ...
        Class DNA-Complex-Formation-bioEngine
            Class bioengine.DNA.U2
            Class bioengine.DNA.U5
            ...
        Class Conform-Change-bioEngine
            Class bioengine.CA1.T2
            ...
        Class Transport-bioEngine
            Class bioengine.EX1.L2
            Class bioengine.M2.EF1
            ...
        ...
    Class Amplifier-bioEngine
        Class Enzyme-bioEngine
            Class bioengine.E1.S1
                Class bioengine.PK1.S1
                Class bioengine.PP1.S1
                Class bioengine.Pase1.S1
                Class bioengine.LK1.S1
                ...
            Class bioengine.E1.S2
            Class bioengine.E1.S1.I1
            Class bioengine.E1.S2.I1
            Class bioengine.E1.S2.I2
            Class bioengine.E3.S1.I1
            ...
        Class Channel-bioEngine
            Class bioengine.CH1.IO1
            Class bioengine.CH1.IO1.L1
            ...
        Class IonPump-bioEngine
            Class bioengine.PU1.IO1
            ...
        ...
    Class Lumped-bioEngine
        Class Complex-Dissoc-bioEngine
        Class Translocation-bioEngine
        Class Auto-Catalitic-bioEngine
```

TABLE 56

Class name	bioengine
Superior class	bionode-object
Attributes specific to class	velocity: rate-constant-sec is 1.0: tau-coeff is given by a tau-coeff-par: bias is given by a bias-par: time-lag is given by a time-delay-par: ph is given by a ~@ph-par: ph-deviation-factor is given by a ph-factor-par: temperature is given by a temperature-par: temp-deviation-factor is given by a temp-factor-par: toggle-state has values show or hide (default is HIDE): type-color
Capabilities and restrictions	none
Class restrictions	unless in administrator, developer, or modeler mode: menu choices exclude additionally: move, name, clone, change-size, color, delete, disable, enable, create-subworkspace, describe, go-to-subworkspace, create-sw: when in modeler mode: menu choices exclude additionally: move, name, disable, enable, create-subworkspace, describe, go-to-subworkspace: unless in administrator or developer mode: attributes visible for bioengine exclude additionally: toggle-state, type-color when in simulation or explorer, mode: attributes visible for bioengine exclude additionally: notes, label:
Menu option	not a final menu choice
Inherited attributes	label is "", with an index: description is ""
Attribute displays	velocity offset by (58, 5)
Stubs	a r-connection port-1 located at top 25: a r-connection port-2 located at top 50: a r-connection port-3 located at top 75: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a e-graph-link gr located at left 11: a e-graph-link ls located at left 31: a e-graph-link up located at right 11: a e-graph-link dn located at right 31
Icon description	bioengine-pattern (width 86; height 42: dark-color = dim-gray, light-color = light-gray, white-color = white, type-color = forest-green

Class name	binding-bioengine
Superior class	bioengine
Attributes specific to class	velocity is given by a binding-rate-pvar
Menu option	not a final menu choice

Class name	amplifier-bioengine
Superior class	bioengine
Attributes specific to class	none
Menu option	not a final menu choice

Class name	lumped-bioengine
Superior class	bioengine
Attributes specific to class	none

TABLE 57

Class name	receptor-bioengine
Superior class	binding-bioengine
Attributes specific to class	none
Menu option	not a final menu choice
Icon description	receptor-bioengine-pattern
Class name	bioengine.r1.l1
Superior class	receptor-bioengine
Attributes specific to class	none
Stubs	a re-connection port-1 located at top 25: a l-connection l1 located at top 50: a p-connection p1 located at bottom 50: a e-graph-link gr located at left 11: a e-graph-link ls located at left 31: a e-graph-link up located at right 11: a e-graph-link dn located at right 31
Class name	enzyme-bioengine
Superior class	amplifier-bioengine
Attributes specific to class	velocity is given by a catalytic-rate-pvar
Icon description	enzyme-bioengine-pattern
Class name	bioengine.e1.s2
Superior class	enzyme-bioengine
Attributes specific to class	none
Stubs	an e-connection port-1 located at top 25: a s-connection s1 located at top 50: a s-connection s2 located at top 75: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a e-graph-link gr located at left 11: a e-graph-link ls located at left 31: a e-graph-link up located at right 11: a e-graph-link dn located at right 31
Class name	complex-dissoc-bioengine
Superior class	lumped-bioengine
Attributes specific to class	velocity is given by a dissociation-rate-pvar
Stubs	a co-connection c1 located at top 25: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a e-graph-link gr located at left 11: a e-graph-link ls located at left 31: a e-graph-link up located at right 11: a e-graph-link dn located at right 31
Icon description	complex-dissoc-bioengine-pattern
Class name	translocation-bioengine
Superior class	lumped-bioengine
Attributes specific to class	velocity is given by a tranlocation-rate-pvar
Stubs	a t-connection t1 located at top 25: a p-connection p2 located at bottom 25: a e-graph-link gr located at left 11: a e-graph-link ls located at left 31: a e-graph-link up located at right 11: a e-graph-link dn located at right 31
Icon description	translocation-bioengine-pattern

TABLE 58

Class name	cell-bioengine
Superior class	bioengine
Attributes specific to class	velocity is given by an activation-rate-pvar
Stubs	a cell-connection ce located at top 25: a l-connection l1 located at top 50: a l-connection l2 located at top 75: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a graph-link gr located at left 11: a graph-link ls located at left 31: a graph-link up located at right 11: a graph-link dn located at right 31
Icon description	cell-bioengine-pattern
Class name	bioengine.ce1.l4
Superior class	cell-bioengine
Attributes specific to class	none
Stubs	a cell-connection ce1 located at top 25: a l-connection l1 located at top 50: a l-connection l2 located at top 75: a l-connection l3 located at top 100: a l-connection l4 located at top 125: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a p-connection p4 located at bottom 100: a p-connection p5 located at bottom 125: a c-graph-link gr located at left 11: a c-graph-link ls located at left 31: a c-graph-link up located at right 11: a c-graph-link dn located at right 31
Icon description	bioengine.ce1.l4-pattern
Class name	bioengine.ce3.12
Superior class	cell-bioengine
Attributes specific to class	none
Stubs	a cell-connection ce1 located at top 25: a cell-connection ce2 located at top 50: a cell-connection ce3 located at top 75: a l-connection l1 located at top 100: a l-connection l2 located at top 125: a p-connection p1 located at bottom 25: a p-connection p2 located at bottom 50: a p-connection p3 located at bottom 75: a p-connection p4 located at bottom 100: a p-connection p5 located at bottom 125: a c-graph-link gr located at left 11: a c-graph-link ls located at left 31: a c-graph-link up located at right 11: a c-graph-link dn located at right 31
Icon description	bioengine.ce1.l4-pattern

TABLE 59

Class name	biorole-object
Superior class	bio-object
Attributes specific to class	stoichiometric-coeff is 1.0
Capabilities and restrictions	none
Class restrictions	unless in administrator..or developer mode: menu choices for biorole-object exclude additionally: move. change-size. describe. create-subworkspace. go-to-subworkspace. rotate-reflect. clone. name: when in simulation. navigation. or general mode: menu choices for biorole-object exclude additionally: transfer. move. name. clone. change-size. color. create-subworkspace. delete. disable, describe. go-to-subworkspace: attributes visible for biorole-object exclude additionally: notes. label: when in simulation mode: menu choices for biorole-object exclude additionally: details. navig-panel: when in navigation or general mode: menu choices for biorole-object exclude additionally: input-panel:
Menu option	a final menu choice
Inherited attributes	label is "", with an index: description is ""
Attribute displays	inherited
Stubs	none
Icon description	biorole-object-pattern (width 100; height 40; dark-color = dim-gray. light-color = light-gray. type-color = sky-blue. white-color = white)

TABLE 60

Class name	bioproduct
Superior class	biorole-object
Attributes specific to class	production-rate is given by a produc-rate-pvar: scaling-precursors is an instance of a symbol-array. with an index
Menu option	a final menu choice
Attribute displays	label offset by (-55. -22): production-rate offset by (-42. 1)
Stubs	a p-connection located at top 30
Icon description	bioproduct-pattern

Class name	cell.bioproduct
Superior class	bioproduct
Attributes specific to class	none
Icon description	bioproduct-pattern

a cell.bioproduct

Names	none
Label	"cell.bioproduct "
Description	""
Stoichiometric coeff	1.0
Production rate	9.9e-99
Scaling precursors	a c-symbol-array

TABLE 61

Class name	bioreactant
Superior class	biorole-object
Attributes specific to class	description is "": alpha-coeff is given by an alpha-coeff-par: contribution is given by a contribution-pvar: consumption-rate is given by a consum-rate-pvar
Capabilities and restrictions	none
Class restrictions	none
Menu option	not a final menu choice
Inherited attributes	stoichiometric-coeff is 1: label is "", with an index
Attribute displays	label offset by (-62, 35): contribution offset by (-42, 14): consumption-rate offset by (-42, 0)
Stubs	none
Icon description	inherited

Class name	cell.bioreactant
Superior class	bioreactant
Attributes specific to class	none
Stubs	a cell-connection located at bottom 30
Icon description	cell.bioreactant-pattern

TABLE 62

Class name	amplifier-bioreactant
Superior class	bioreactant
Attributes specific to class	effective-binding-sites is 1.0
Menu option	not a final menu choice
Attribute displays	inherited
Stubs	inherited
Icon description	inherited

Class name	enzyme.r
Superior class	amplifier-bioreactant
Attributes specific to class	catalytic-rate-constant is 1.0: scaled-cat-rate.k is 1.0:
Stubs	an e-connection located at bottom 30
Icon description	bioreactant-pattern (width 100; height 40; dark-color = dim-gray, light-color = light-gray, type-color = red, white-color = white)

Class name	prot.kinase.r
Superior class	enzyme.r
Attributes specific to class	none
Stubs	inherited
Icon description	bioreactant-pattern (width 100; height 40; dark-color = dim-gray, light-color = light-gray, type-color = coral, white-color = white)

a prot.kinase.r

Names	none
Label	"prot.kinase"
Description	""
Stoichiometric coeff	1.0
Alpha coeff	1.0
Contribution	9.9e-99
Consumption rate	9.9e-99
Effective binding sites	1.0
catalytic-rate-constant	1.0
scaled-cat-rate.k	1.0

TABLE 63

Class name	source-bioreactant
Superior class	bioreactant
Attributes specific to class	none
Menu option	not a final menu choice

Class name	substrate.r
Superior class	source-bioreactant
Attributes specific to class	michaelis-constant is 9.9e-99
Stubs	a s-connection s1 located at bottom 30
Icon description	substrate.r-pattern

Class name	leading-bioreactant
Superior class	bioreactant
Attributes specific to class	effective-binding-sites is 1
Icon description	leading-bioreactant-pattern

Class name	cell-receptor.r
Superior class	leading-bioreactant
Attributes specific to class	none
Stubs	a rc-connection r located at bottom 30
Icon description	cell-receptor.r-pattern

Class name	binding-bioreactant
Superior class	bioreactant
Attributes specific to class	equilibrium-dissociation-constant is 9.9e-99 scaled-equil.dissoc.k is 1.0
Menu option	not a final menu choice

Class name	ligand.r
Superior class	binding-bioreactant
Attributes specific to class	none
Stubs	a l-connection l1 located at bottom 30
Icon description	ligand.r-pattern

Class name	extracell-ligand.r
Superior class	binding-bioreactant
Attributes specific to class	none
Stubs	a l-connection l1 located at bottom 30
Icon description	bioreactant-pattern (width 100; height 40; dark-color = dim-gray, light-color = light-gray, type-color = pale-green, white-color = white)

an extracell-ligand.r

Names	none
Label	"extracell-ligand"
Description	""
Stoichiometric coeff	1.0
Alpha coeff	1.0
Contribution	9.9e-99
Consumption rate	9.9e-99
Equilibrium dissociation constant	9.9e-99
Scaled equil.dissoc.k	1.0

TABLE 64

Class name	single-bioreactant
Superior class	bioreactant
Attributes specific to class	none
Menu option	not a final menu choice
Class name	translocated.r
Superior class	single-bioreactant
Attributes specific to class	none
Stubs	a t-connection located at bottom 30
Icon description	translocated.r-pattern
Class name	complex.r
Superior class	single-bioreactant
Attributes specific to class	none
Stubs	a co-connection u1 located at bottom 30
Icon description	translocated.r-pattern
Class name	inhibitor.bioreactant
Superior class	bioreactant
Attributes specific to class	inhibition-constant is 9.9e-99; scaled-inhibitor.k is 1.0
Stubs	an i-connection i1 located at bottom 30
Icon description	inhibitor.bioreactant-pattern
Class name	comp-inhibitor.r
Superior class	inhibitor.bioreactant
Attributes specific to class	none
Icon description	comp-inhibitor.r-pattern
Class name	antagonist.r
Superior class	inhibitor.bioreactant
Attributes specific to class	none
Stubs	an an-connection an1 located at bottom 30
Icon description	antagonist.r-pattern

TABLE 65

Class name	biorole-post
Superior class	biopost
Attributes specific to class	ref-bioreservoir is none
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: move. clone. delete. rotate-reflect. color. describe. create-subworkspace: when in modeler mode: menu choices include additionally: delete: menu choices exclude additionally: show-br: when in navigation or simulation mode: selecting any biopost implies show-br: when in explorer mode: selecting any biopost implies activate-br
Inherited attributes	toggle-state has values show or hide (default is HIDE)
Stubs	a icon-wire located at right 7 with style diagonal
Icon description	biorole-post-pattern (width 8: height 16)
Class name	bioreactant-post
Superior class	biorole-post
Attributes specific to class	none
Stubs	a icon-wire located at right 7
Icon description	bioreactant-post-pattern (width 8: height 16)
Class name	bioproduct-post
Superior class	biorole-post
Attributes specific to class	none
Stubs	a icon-wire located at left 7
Icon description	bioproduct-post-pattern (width 8: height 16)

TABLE 66

Class name	biomodel
Superior class	bioview-object
Attributes specific to class	none
Class restrictions	unless in administrator. uil-build. or developer mode: menu choices for biomodel exclude additionally: change-size: attributes visible for biomodel exclude additionally: toggle-state: when in simulation. navigation. or explorer mode: selecting any biomodel implies details
Menu option	not a final menu choice
Inherited attributes	references is "": warnings is "": toggle-state has values show or hide (default is HIDE): label is "". with an index: description is ""
Attribute displays	inherited
Stubs	inherited
Icon description	biomodel-pattern (width 100: height 100: dark-color = dim-gray. light-color = light-gray. white-color = white)
Class name	sub-model
Superior class	biomodel
Attributes specific to class	none
Menu option	a final menu choice
Inherited attributes	references is "": warnings is "": toggle-state has values show or hide (default is HIDE). with an index: label is "". with an index: description is ""
Attribute displays	label offset by (-73. 15)
Stubs	inherited
Icon description	sub-model-pattern (width 160: height 60: dark-color = dim-gray. light-color = light-gray. white-color = white. status-color = medium-aquamarine. flag-color = medium-aquamarine:
Class name	bioreservoir-bin
Superior class	biomodel
Attributes specific to class	none
Menu option	a final menu choice
Icon description	bioreservoir-bin-pattern (width 100: height 60: dark-color = dim-gray. light-color = light-gray. white-color = white. type-color = gray. status-color = cadet-blue:
Class name	on-hold-bin
Superior class	biomodel
Attributes specific to class	none
Capabilities and restrictions	activatable-subworkspace
Menu option	a final menu choice
Icon description	on-hold-bin-pattern (width 50: height 60: dark-color = dim-gray. light-color = light-gray. white-color = white. status-color = green-yellow:

TABLE 67

Class name	cell-compartment
Superior class	biomodel
Attributes specific to class	surface-or-volume is given by a compartment-volume-par
Icon description	cell-compartment-pattern (width 90; height 90; dark-color, light-color, white-color)
Class name	cell-surface
Superior class	cell-compartment
Attributes specific to class	internal-markers is ""
Capabilities and restrictions	none
Menu option	a final menu choice
Icon description	cell-surface-pattern (width 130; height 200; status-color, flag-color;
Class name	nucleus
Superior class	cell-compartment
Attributes specific to class	none
Menu option	a final menu choice
Icon description	nucleus-pattern (width 90; height 90; status-color, flag-color)
Class name	cell-membrane
Superior class	cell-compartment
Attributes specific to class	none
Menu option	a final menu choice
Icon description	cell-membrane-pattern (width 90; height 90; status-color , flag-color)
Class name	membrane-compartment
Superior class	cell-membrane
Attributes specific to class	none
Menu option	a final menu choice
Icon description	membrane-compartment-pattern (width 90; height 90; status-color, flag-color)

TABLE 68

Class name	top-container
Superior class	biomodel
Attributes specific to class	volume is given by a compartment-volume-par
Menu option	not a final menu choice
Inherited attributes	references is ""; warnings is ""; toggle-state has values show or hide (default is HIDE) label is "", with an index; description is ""
Class name	cell-biomodel
Superior class	top-container
Attributes specific to class	max-volume is given by a cell-volume-par; volume is given by a cell-volume-par; cell-radius is given by a cell-radius-par; species is ""; organ-or-system is "" tissue is ""; cell-lineage; phenotype is ""; cell-cycle-duration is given by a cycle-time-pvar; reg-factors is ""; expr-receptors is ""; secr-factors is ""; total-protein is given by a total-protein-par generation-number is given by a generation-number-par; number-of-cells is given by a cell-number-par; mass-per-cell is given by a cell-mass-par cell-phase is G0
Attribute displays	label offset by (-55, -52)
Icon description	cell-biomodel-pattern (width 100; height 100; dark-color, light-color, white-color, type-color, status-color)

TABLE 69a

Class name	time-compartment
Superior class	biomodel
Attributes specific to class	next-activation-block: deactivation-block: activation-hold-interval is given by a time-delay-par: deactivation-time is "": deactivation-hold-interval is given by a time-delay-par: status has values activated or deactivated (default is DEACTIVATED)
Capabilities and restrictions	activatable subworkspace
Menu option	not a final menu choice
Stubs	an output cycle-path next-1 located at right 40: an input cycle-path previous-1 located at left 40
Icon description	time-compartment-pattern (width 80: height 80: dark-color. light-color. white-color)

Class name	time-submodel
Superior class	time-compartment
Attributes specific to class	none
Stubs	an output cycle-path next-1 located at bottom 80: an input cycle-path previous located at top 80
Icon description	time-submodel-pattern (width 160: height 60: status-color. flag-color)

Class name	cell-phase
Superior class	time-compartment
Attributes specific to class	typical-fraction: generation-number is given by a generation-number-par: number-of-cells is given by a cell-number-par: mass-per-cell is given by a cell-mass-par
Menu option	not a final menu choice
Icon description	cell-phase-pattern (width 80: height 80)

Class name	g0-phase
Superior class	cell-phase
Attributes specific to class	none
Stubs	an output cycle-path g1 located at bottom 20: an input cycle-path m located at right 60: an input cycle-path di located at left 40
Icon description	g0-phase-pattern (width 160: height 80: status-color. type-color. flag-color)

TABLE 69b

Class name	timer
Superior class	biotool
Attributes specific to class	toggle-state has values show or hide (default is HIDE): activation-time is given by an elapsed-simul-time-par
Class restrictions	unless in administrator or uil-build mode: menu choices for timer exclude additionally: move. name. change-size. color. describe. go-to-subworkspace: when in modeler. simulation. navigation. or explorer mode: menu choices for timer include: create-sw. show- sw. hide-sw
Icon description	timer-pattern (width 30: height 30): dark-color. light-color.

Class name	count-down-timer
Superior class	timer
Attributes specific to class	count-down is given by a timer-count-down-par

Class name	count-up-timer
Superior class	timer
Attributes specific to class	count-up is given by a timer-count-up-par

TABLE 70

Class name	biomodel-post
Superior class	biopost
Attributes specific to class	ref-biomodel is "inferred"
Class restrictions	when in navigation or simulation mode: selecting any biomodel-post implies show-bm
Icon description	biomodel-post-pattern (width 25; height 35)

Class name	output-bm-post
Superior class	biomodel-post
Attributes specific to class	none

an user-menu-choice

Label	show-bm
Applicable class	biomodel-post
Condition	none
Action	start SHOW-BM-PROC (the item, this window)

SHOW-BM-PROC (post: class biomodel-post , win: class window)
  BM = the biomodel named by the ref-biomodel of post;
  case the toggle-state of post is
    show:
      change the color-pattern of post so that light-color is dim-gray, dark-color is light-gray ;
      show the WS of BM on win at the center of the screen; conclude that the toggle-state of post is hide;
      change the color-pattern of post so that lamp = the symbol maroon;
    hide:
      change the color-pattern of post so that light-color is light-gray, dark-color is dim-gray ;
      hide the WS of BM on win; conclude that the toggle-state of post is show;
      change the color-pattern of post so that lamp = the symbol green;
  return

TABLE 71

Class name	cell-interaction
Superior class	biomodel
Attributes specific to class	none
Inherited attributes	references is ""; warnings is ""; toggle-state has values show or hide (default is HIDE); label is "", with an index; description is ""
Icon description	cell-interaction-pattern (width 140; height 100; dark-color, light-color, white-color, type-color-1, type-color-2, status-color-1, status-color-2)

Class name	interacting-cell
Superior class	biomodel
Attributes specific to class	master-cell-biomodel is none
Class restrictions	unless in administrator or developer mode, menu choices exclude additionally: details, move, name, disable, describe; attributes visible exclude additionally: notes, toggle-state, names; when in simulation, navigation, or explorer mode: selecting any interacting-cell implies master-details
Attribute displays	label offset by (-55, -52)
Stubs	a c-link located at right 10; a c-link located at right 30; a c-link located at right 50; a c-link located at right 70; a c-link located at right 90; a c-link located at bottom 30; a c-link located at bottom 70; a c-link located at left 10; a c-link located at left 30; a c-link located at left 50; a c-link located at left 70; a c-link located at left 90; a c-link located at top 20; a c-link located at top 80
Icon description	interacting-cell-pattern (width 100; height 100; dark-color, light-color, white-color, type-color, status-color)

TABLE 72a

Class name	extracellular-component
Superior class	biopost
Attributes specific to class	label is "": ref-bioentity is none: ref-bioreservoir is none: ref-bioprocess is none: description is ""
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: rotate-reflect. disable. describe. create-subworkspace: when in modeler or general mode: menu choices exclude additionally: show-bp. show-br: when in simulation or navigation mode: menu choices exclude additionally: bp-activation. br-activation: when in simulation. navigation. or general mode: menu choices exclude additionally:delete. move. name. clone. change-size. color. unname-it. set-refs: attributes visible exclude additionally:names. ref-bioreservoir. ref-bioprocess. ref-bioenuty
Menu option	not a final menu choice
Attribute displays	label offset by (-60. 6)
Stubs	a link located at right 10: a link located at left 10
Icon description	extracellular-component-pattern (width 120: height 20: white-color = white)
Class name	surface-component
Superior class	extracellular-component
Attributes specific to class	surface-expression-info is ""
Icon description	surface-component-pattern (width 120: height 20: white-color = white)
Class name	secreted-component
Superior class	extracellular-component
Attributes specific to class	secretion-info is ""
Icon description	secreted-component-pattern (width 120: height 20: white-color = white)
Class name	systemic-component
Superior class	extracellular-component
Attributes specific to class	none
Icon description	systemic-component-pattern (width 120: height 20: white-color = white)

TABLE 72b

	an user-menu-choice
Label	set-refs
Applicable class	extracellular-component
Condition	the item has a name
Action	start EXTRACELL-COMP-SET-REFS-PROC (the item. this window)

EXTRACELL-COMP-SET-REFS-PROC (post: class extracellular- component. W: class window)
PP = the biopool-post named by the name of post:
BR = the bioreservoir superior to the workspace of PP:
RP = the biorole-post named by the name of post:
BP = the bioprocess superior to the workspace of RP:
conclude that the ref-bioreservoir of post = the name of BR:
conclude that the ref-bioprocess of post = the name of BP:
if the ref-bioentity of BR is not none then conclude that the ref-bioentity of post = the ref-bioentity of BR:
make post permanent:

TABLE 72c

	an user-menu-choice
Label	show-bp
Applicable class	extracellular-component
Condition	the item has a name and the ref-bioprocess of the item is not none
Action	start EXTRACELL-COMP-SHOW-BP-PROC (the item. this window)

```
EXTRACELL-COMP-SHOW-BP-PROC (post: class extracellular-component . W: class window)
  BP = the bioprocess named by the ref-bioprocess of post:
  view = the subworkspace of BP
  change the size of view to minimum:
  show  view on W 100 units above the center of the screen:
```

TABLE 72d

	an user-menu-choice
Label	show-br
Applicable class	extracellular-component
Condition	the item has a name and the ref-bioreservoir of the item is not none
Action	start EXTRACELL-COMP-SHOW-BR-PROC (the item. this window)

```
EXTRACELL-COMP-SHOW-BR-PROC (post: class extracellular-component. W: class window)
  BR = the bioreservoir named by the ref-bioreservoir of post:
  view = the subworkspace of BP
  change the size of view to minimum:
  show view on W 100 units below the center of the screen:
```

TABLE 72e

	an user-menu-choice
Label	bioentity
Applicable class	extracellular-component
Condition	the ref-bioentity of the item is not none
Action	start EXTRACELL-COMP-BIOENTITY-PROC (theitem. this window)

```
EXTRACELL-COMP-BIOENTITY-PROC (EC: class extracellular-  component. W: class window)
  BET = the bioentity named by the ref-bioentity of EC :
  ESW = the subworkspace of BET:
  if the toggle-state of EC is hide then:
    conclude that the toggle-state of EC is show:
    change the size of ESW to minimum:
    show ESW on W at the center of the screen:
  else if the toggle-state of EC is show then
    hide ESW on W:
    conclude that the toggle-state of EC is hide:
```

TABLE 72f

	an user-menu-choice
Label	unname-it
Applicable class	extracellular-component
Condition	the item has a name
Action	start EXTRACELL-COMP-REMOVE-NAME-PROC (the item. this window)

```
EXTRACELL-COMP-REMOVE-NAME-PROC (post: class biorole-post. W: class window)
  BP = the bioprocess superior to the workspace of post:
  if there exists a biopool connected to post then:
    BR = the bioreservoir superior to the workspace of the biopool connected to post:
    if there is a biopool-post named by the name of post then
      PP = the biopool-post named by the name of post:
      change the ref-bioprocess of PP to "none":
      make PP permanent:
  change the ref-bioreservoir of post to "none":
  change the names of post to "none":
  make post permanent:
```

TABLE 73

Class name	organ
Superior class	biomodel
Attributes specific to class	none
Menu option	not a final menu choice
Icon description	organ-pattern (width 140: height 100: type-color, status-color)
Class name	lymph-node
Superior class	organ
Attributes specific to class	cell-number is given by a cell-number-par: b.lymphocytes is given by a cell-number-par: t.lymphocytes is given by a cell-number-par: macrophages is given by a cell-number-par: b..cd4..cd8 is n.n.n: terp.cd4p.cd8n is given by a phenotype-par: terp.cd4n.cd8p is given by a phenotype-par: other-phenotypes
Menu option	a final menu choice
Icon description	lymph-node-pattern
Class name	organ-region
Superior class	biomodel
Attributes specific to class	none
Menu option	not a final menu choice
Icon description	organ-region-pattern
Class name	differentiation-pathway
Superior class	biomodel
Attributes specific to class	none
Icon description	differentiation-pathway-pattern (width 100: height 120: dark-color, light-color, white-color)
Class name	cell
Superior class	biomodel
Attributes specific to class	in-tissues is "":
Attribute displays	label offset by (35, 30)
Stubs	a c-link located at top 40: a c-link located at bottom 20: a c-link located at bottom 40: a c-link located at bottom 60
Icon description	cell-pattern (width 80: height 80: dark-color, light-color, lineage-color)

TABLE 74

a tabular function of 1 argument

Table of values

[Substrate1]	[Product1]([Substrate1])
0	112
3.13	757
6.25	1.305
12.50	1.932
25.00	3.045
50.00	3.356
100.00	3.478

TABLE 76

CHAINING RULES

Options for all whenever rules: not invocable via backward chaining.
not invocable via forward chaining.
may cause data seeking.
may cause forward chaining

---

for any bioprocess BP
for any bioprocess BP1 that is *an-upstream-bioprocess-of* BP
for any bioprocess BP2 that is *an-upstream-bioprocess-of* BP
whenever BP1 ceases to be *an-upstream-bioprocess-of* BP and when not (there exists a bioprocess BP3 such that (BP2 is *an-upstream-bioprocess-of* BP3 and BP3 is *an-upstream-bioprocess-of BP*)) then
conclude that BP2 is not *A-DOWNSTREAM-BIOPROCESS-OF* BP

---

for any bioprocess BP
for any bioprocess BP1 that is *an-upstream-bioprocess-of* BP
whenever any bioprocess BP2 becomes *an-upstream-bioprocess-of* BP1 and when BP2 is not *an-upstream-bioprocess-of* BP then
conclude that BP2 is *AN-UPSTREAM-BIOPROCESS-OF* BP

---

for any bioprocess BP
for any bioprocess BP1 that is *a-downstream-bioprocess-of* BP
for any bioprocess BP2 that is *a-downstream-bioprocess-of* BP
whenever BP1 ceases to be *a-downstream-bioprocess-of* BP and when not (there exists a bioprocess BP3 such that (BP2 is *a-downstream-bioprocess-of* BP3 and BP3 is *a-downstream-bioprocess-of* BP)) then
conclude that BP2 is not *A-DOWNSTREAM-BIOPROCESS-OF* BP

---

for any bioprocess BP
for any bioprocess BP1 that is *a-downstream-bioprocess-of* BP
whenever any bioprocess BP2 becomes *a-downstream-bioprocess-of* BP1 and when BP2 is not *a-downstream-bioprocess-of* BP then
conclude that BP2 is *A-DOWNSTREAM-BIOPROCESS-OF* BP

TABLE 77

INFERENCE BLOCK RULES

for any event-deactivation-block B that is an attribute of any time-compartment CC
for any time-compartment NC connected at the next-1 of CC
whenever the status S of NC receives a value and when S is activated then
deactivate the subworkspace of CC and conclude that the status of CC is deactivated

---

whenever the time-interval I of any time-activation-block B receives a value and when the beta-coeff of B * I >= the time-threshold of B then
conclude that the outcome of B is above-threshold

---

whenever the time-interval I of any time-deactivation-block B receives a value and when the beta-coeff of B * I >= the time-threshold of B then
conclude that the outcome of B is above-threshold

---

whenever the checkpoint C of any checkpoint-to-floor-block B receives a value and when the beta-coeff of B * C <= the threshold of B then
conclude that the outcome of B is below-threshold

---

whenever the checkpoint C of any checkpoint-to-ceiling-block B receives a value and when the beta-coeff of B * C >= the threshold of B then
conclude that the outcome of B is above-threshold

TABLE 78

CELL CYCLE RULES

for any g0-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then
activate the subworkspace of the g1-phase G connected at the g1 of P and conclude that the status of G is activated

---

for any g1-phase P
whenever the outcome O of the differentiation-activation-block of P receives a value and
when the status of the s-phase connected at the s of P is deactivated and O is not within-threshold then
activate the subworkspace of the differ-stage N connected at the di of P and conclude that the status of N is activated

---

for any g1-phase P
whenever the outcome O of the next-activation-block of P receives a value and when the status of the differ-stage connected at the di of P is deactivated and O is not within-threshold then
activate the subworkspace of the s-phase N connected at the s of P and conclude that the status of N is activated

---

for any s-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then
activate the subworkspace of the g2-phase G connected at the g2 of P and conclude that the status of G is activated

---

for any g2-phase P
whenever the outcome O of the apoptosis-activation-block of P receives a value and when
the status of the m-phase connected at the m of P is deactivated and O is not within-threshold then
activate the subworkspace of the apoptosis N connected at the ap of P and conclude that the status of N is activated

---

for any g2-phase P
whenever the outcome O of the next-activation-block of P receives a value and when the status of the apoptosis connected at the ap of P is deactivated and O is not within-threshold then
activate the subworkspace of the M-phase N connected at the m of P and conclude that the status of N is activated

---

for any M-phase P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then
conclude that the generation-number N of the g0-phase G connected at the g0 of P = N + 1 and conclude that the number-of-cells C of G = 2 * C and conclude that the mass-per-cell of G = the mass-per-cell of P / 2 and deactivate the subworkspace of P and conclude that the status of P is deactivated

---

for any M-phase P
whenever the outcome O of the g1-activation-block of P receives a value and when O is not within-threshold then
conclude that the generation-number N of the g0-phase G connected at the g0 of P = N + 1 and conclude that the number-of-cells C of G = 2 * C and conclude that the mass-per-cell of G = the mass-per-cell of P / 2 and activate the subworkspace of the g1-phase G connected at the g1 of P and conclude that the status of G is activated and deactivate the subworkspace of P and conclude that the status of P is deactivated

---

for any differ-stage P
whenever the outcome O of the next-activation-block of P receives a value and when O is not within-threshold then
activate the subworkspace of the g0-phase G0 connected at the ng of P and conclude that the status of G0 is activated and activate the subworkspace of the g1-phase G1 connected at the ng of P and conclude that the status of G1 is activated

---

TABLE 79

ICON COLOR RULES

for any bioview-object O
whenever the toggle-state T of O receives a value and when T is hide then
change the color-pattern of O so that light-color is light-gray, dark-color is dim-gray, white-color is white

---

for any bioprocess P
for any bioengine E
whenever the type-color of E receives a value then
change the color-pattern of P so that type-color = the type-color of E

---

for any bioprocess BP
whenever BP is moved by the user and when the subworkspace SW of BP exists and there exists a bioengine E upon SW and not (the type-color of BP = the type-color of E) then
conclude that the type-color of BP = the type-color of E

---

for any bionode-object O
whenever the toggle-state T of O receives a value and when T is hide then
change the color-pattern of O so that light-color is light-gray, dark-color is dim-gray, white-color is white

---

for any complex-bioentity O
whenever the toggle-state T of O receives a value and when T is show then
change the color-pattern of O so that light-color is dim-gray, dark-color is light-gray, white-color is black

---

for any bioentity-notes N
whenever the toggle-state T of N receives a value and when T is hide then
change the color-pattern of N so that light-color is light-gray, dark-color is dim-gray, white-color is white

---

for any bioentity-notes N
whenever the toggle-state T of N receives a value and when T is show then
change the color-pattern of N so that light-color is dim-gray, dark-color is light-gray, white-color is black

---

for the subworkspace SW of any bioreservoir-bin BRB
for any bioreservoir BR upon SW
whenever BR is moved by the user and when not (there exists a bioreservoir upon SW) then
change the type-color of BRB to gray

---

for the subworkspace SW of any bioreservoir-bin BRB
for any bioreservoir BR upon SW
whenever BR is moved by the user and when there exists a bioreservoir upon SW then
change the type-color icon-color of BRB to purple

---

for any experiment-selection ES
whenever ES is moved by the user and when the currently-selected of ES is false then
change the type-color icon-color of ES to goldenrod

---

for any experiment-selection ES
whenever ES is moved by the user and when the currently-selected of ES is true then
change the type-color icon-color of ES to gold

TABLE 80

LABELING RULES

for any bioentity BET
whenever BET is moved by the user and when BET has a name then
conclude that the master-bioentity of BET = the name of BET

---

for any bioentity BET
for any bio-object-title T upon the subworkspace of BET
whenever BET is moved by the user then
conclude that the label of T = "[the label of BET]"

---

for any bioentity-notes FN upon the SW of any bioentity BET
whenever FN is moved by the user then
conclude that the label of the bio-object-title upon the subworkspace of FN = the label of BET

TABLE 81

ICON MOVEMENT RULES

for any biopool P connected to any graph-tracer
whenever P is moved by the user then
move the scaled-br-graph-tracer connected to P to (left-edge(P) - 8, middle-y(P) + 11) and
move the absolute-br-graph-tracer connected to P to (left-edge(P) - 8, middle-y(P) - 10) and
move the br-chaininig-tracer connected to P to (right-edge(P) + 6, middle-y(P) + 11) and
move the query-tracer connected to P to (right-edge(P) - 6, middle-y(P) - 10) and
change the core stripe-color of every graph-link connected to P to transparent

---

for any bioengine E connected to any graph-tracer
whenever E is moved by the user then
move the scaled-bp-graph-tracer connected to E to (left-edge(E) - 8, middle-y(E) + 11) and
move the bp-chaininig-tracer connected to E to (left-edge(E) - 8, middle-y(E) - 10) and
move the up-path-tracer connected to E to (right-edge(E) + 8, middle-y(E) + 11) and
move the down-path-tracer connected to E to (right-edge(E) + 8, middle-y(E) - 10) and
change the core stripe-color of every graph-link connected to E to transparent

---

for any bioproduct P connected to any bioproduct-post PP
whenever P is moved by the user then
move PP to (right-edge(P) + 4, middle-y(P))
change the core stripe-color of the icon-wire connected to R to transparent and
change the core stripe-color of the icon-wire connected to P to transparent

---

for any bioreactant R connected to any bioreactant-post RP
whenever R is moved by the user then
move RP to (left-edge(R) - 3, middle-y(R)) and
change the core stripe-color of the icon-wire connected to R to transparent

---

TABLE 82

---

the VELOCITY of any BIOENGINE.E1.S1.I1 E = the first of the following expressions that has a value (the effective-binding-sites of the enzyme.r Z connected at the port-1 of E * the catalytic-rate-constant of Z * (the concentration of the soluble-mol-pool connected to Z * the concentration SQ of the soluble-mol-pool connected to the substrate.r S connected at the s1 of E /
(if not (there exists an inhibitor.bioreactant connected at the i1 of E)
then SQ + the michaelis-constant of S else
(if there exists a comp-inhibitor.r CI connected at the i1 of E then SQ +
the michaelis-constant of S * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the inhibition-constant of CI) else
(if there exists a noncomp-inhibitor.r NI connected at the i1 of E then
SQ * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the inhibition-constant of NI + the michaelis-constant of S) * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the Inhibition-constant of NI) else
(if there exists an uncomp-inhibitor.r UI connected at the i1 of E then
SQ * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the inhibition-constant of UI + the michaelis-constant of S) else
(if there exists a mixed-inhibitor.r MI connected at the i1 of E then
SQ * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the inhibition-constant of MI + the michaelis-constant
of S) * (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I / the inhibition-constant of MI))))))). 0.0)

TABLE 83

GENERIC-SIMULATION-FORMULAS for Scaled-Valued Variables

the SCALED-AMOUNT C of any BIOPOOL P() = (max (9.9e-9, the accumulation of P()))

state variable : d / dt (the ACCUMULATION C of any BIOPOOL P()) = the input-rate of P() - the output-rate of P() + the scaled-entry of P() - the decay-rate-factor of the bioreservoir superior to the workspace of P() * the scaled-amount of P(), with initial value the scaled-basal-amount of the bioreservoir superior to he workspace of P()

the INPUT-RATE of any BIOPOOL P() = the first of the following expressions that has a value ((max (0.0, the sum over each bioproduct P connected to P() of (the production-rate of P) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the scaled-input-model-box connected to P(), 0.0))) , 9.9e-9) .

the OUTPUT-RATE of any BIOPOOL P() = the first of the following expressions that has a value (max (0.0, the sum over each bioreactant R connected to P() of (the consumption-rate of R) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the output-model-box connected to P(), 0.0)) , 9.9e-9)

the CONSUMPTION-RATE of any BIOREACTANT R = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to R / the stoichiometric-coeff of R), 9.9e-9)

the CONSUMPTION-RATE of any AMPLIFIER-BIOREACTANT R = 0.0

the PRODUCTION-RATE of any BIOPRODUCT P = the first of the following expressions that has a value (max (0.0, the velocity of the bioengine connected to P * the stoichiometric-coeff of P), 9.9e-9)

the VELOCITY of any BIOENGINE E = the first of the following expressions that has a value (max (0.0, the product over each bioreactant R connected to E of (the alpha-coeff of R * the contribution of R) ^ the rate-constant-sec of E * the tau-coeff of E + the bias of E), 9.9e-9)

TABLE 84

GENERIC-SIMULATION-FORMULAS FOR THE CONTRIBUTION
Based on Scaled-Valued Variables and Parameters

the CONTRIBUTION of any AMPLIFIER-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the scaled-amount of the biopool connected to R)), 0.5)

the CONTRIBUTION of any SUBSTRATE.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool P() connected to R / (the scaled-michaelis.k of R + Q))) , 0.5)

the CONTRIBUTION of any INHIBITOR.BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool P() connected to R / (the scaled-inhibition.k of R + Q))) , 0.5)

the CONTRIBUTION of any LEADING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the scaled-amount of the biopool connected to R)) ^ the stoichiometric-coeff of R, 0.5)

the CONTRIBUTION of any BINDING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool P() connected to R / (the scaled-equil.dissoc.k of R + Q))) ^ the stoichiometric-coeff of R , 0.5)

the CONTRIBUTION of any ION.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount Q of the biopool P() connected to R / (the scaled-equilibrium.k of R + Q))) , 0.5)

the CONTRIBUTION of any SINGLE-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the scaled-amount of the biopool connected to R)), 0.5)

TABLE 85

GENERIC SIMULATION FORMULAS FOR ABSOLUTE-VALUED VARIABLES

the CONCENTRATION of any SOLUBLE-MOL-POOL PO = (max (0.0. the accumulation of PO))

---

the DENSITY of any COMPLEXED-MOL-POOL PO = (max (0.0. the accumulation of PO))

---

state variable : d / dt (the ACCUMULATION of any SOL-MOL-POOL PO) = the input-rate of PO - the output-rate of PO + the concentration-entry of PO - the decay-rate-factor of the sol-mol-reservoir superior to the workspace of PO * the concentration of PO. with initial value the basal-concentration of PO

---

state variable : d / dt (the ACCUMULATION of any COMPLEXED-MOL-POOL PO) = the input-rate of PO - the output- rate of PO + the density-entry of PO - the decay-rate- factor of the bound-mol-reservoir superior to the workspace of PO * the density of PO. with initial value the basal-density of PO

---

the INPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value (max (0.0. the sum over each bioproduct P connected to PO of (the production-rate of P) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the input-model-box connected to PO. 0.0)). 0.0)

---

the OUTPUT-RATE of any BIOPOOL PO = the first of the following expressions that has a value (max (0.0. the sum over each bioreactant R connected to PO of (the consumption-rate of R) + the first of the following expressions that has a value (the output-1 of the model-block M upon the subworkspace of the output-model-box connected to PO. (0.0)) . 0.0)

---

the PRODUCTION-RATE of any BIOPRODUCT P = the first of the following expressions that has a value (max (0.0. the velocity of the bioengine connected to P * the stoichiometric-coeff of P). 0.0)

---

the CONSUMPTION-RATE of any BIOREACTANT R = the first of the following expressions that has a value (max (0.0. the velocity of the bioengine connected to R / the stoichiometric-coeff of R). 0.0)

---

the CONSUMPTION-RATE of any AMPLIFIER-BIOREACTANT R = 0.0

---

the VELOCITY of any BIOENGINE.E1.S1.I1 E = the first of the following expressions that has a value (the effective-binding-sites of the enzyme.r Z connected at the port-1 of E * the catalytic-rate-constant of Z * (the concentration of the sol-mol-pool connected to Z * the concentration SQ of the sol-mol-pool connected to the substrate.r S connected at the s1 of E / SQ + the michaelis-constant of S * (if there exists a comp-inhibitor.r I connected at the i1 of E then (1 + the concentration of the sol-mol-pool connected to I / the Inhibition-constant of I) else 1)). 1.0e-99)

TABLE 86

GENERIC SIMULATION FORMULAS FOR MIXED-TYPE SIMULATIONS

state variable : d / dt (the ACCUMULATION of any BIOPOOL PO) = the input-rate of PO - the output-rate of PO + the density-entry of PO - the decay-rate-factor of the bioreservoir superior to the workspace of PO * the density of PO, with initial value the basal-density of PO

---

the DENSITY of any BIOPOOL PO = the first of the following expressions that has a value (max (1.0e-9, (the scaled-amount of PO * the scaling-density of PO)), 9.9e-99)

---

the CONCENTRATION C of any BIOPOOL PO = the first of the following expressions that has a value (max (1.0e-9, (the density of PO / (6.023e23 * top-volume-po(PO)))) , 9.9e-99)

---

the CONTRIBUTION of any AMPLIFIER-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the density of the biopool PO connected to R / the scaling-density of PO)), 0.5)

---

the CONTRIBUTION of any SUBSTRATE.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R / (the michaelis-constant of R + Q))) , 0.5)

---

the CONTRIBUTION of any INHIBITOR.BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R / (the inhibition-constant of R + Q))) , 0.5)

---

the CONTRIBUTION of any LEADING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the effective-binding-sites of R * the density of the biopool PO connected to R / the scaling-density of PO)) ^ the stoichiometric-coeff of R, 0.5)

---

the CONTRIBUTION of any BINDING-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the density Q of the biopool PO connected to R / (the equilibrium-dissociation-constant of R + Q))) ^ the stoichiometric-coeff of R , 0.5)

---

the CONTRIBUTION of any ION.R R = the first of the following expressions that has a value (min (1.0, max (0.0, the concentration Q of the biopool PO connected to R / (the equilibrium-constant of R + Q))) , 0.5)

---

the CONTRIBUTION of any SINGLE-BIOREACTANT R = the first of the following expressions that has a value (min (1.0, max (0.0, the density of the biopool PO connected to R / the scaling-density of PO)), 0.5)

TABLE 87

GENERIC SIMULATION FORMULAS for Model Blocks

the output-1 of any CONSTANT.F CF = the constant of CF
Tracing and breakpoints default the output-1 of any INCREASING.F = the current time * 1e-5
the output-1 of any PROPORTIONAL.F PF = the input-1 of PF * the gain of PF + the bias of PF
the output-1 of any SIGMOID.F SF = 1.0 / (1.0 + exp (- the gain of SF * (10 * the input-1 of SF) - the bias of SF))
the output-1 of any EXP.C.PDF C = 1.0 - exp (- the x-val of E / the mean of E)
the output-1 of any EXP.I.PDF E = exp (- the x-val X of E / the mean M of E) / M
the output-1 of any UNIF.C.PDF UC = (if the x-val X of UC < the min-val MI of UC then MI else (if X > the max-val MA of UC then MA else ((X - MI) / (MA - MI))))
the output-1 of any UNIF.RN.F UR = random (the min-val of UR, the max-val of UR)
the output-1 of any NORMAL.RN.PDF NR = the mean of NR + ((the max-val of NR + the min-val of NR) / 6) * sqrt (-2.0 * ln (random (0.0, 1.0))) * sin (2.0 * pi () * random (0.0, 1.0)) * ln (random (0.0, 1.0))
the output-1 of any NORMAL.I.PDF NI = exp ((expt ((the x-val of NI - the mean of NI), 2)) / (2.0 * (the max-val MA of NI + the min-val MI of NI) / 6)) * (1.0 / (sqrt (2.0 * pi () * (MA + MI) / 6)))
the output-1 of any SIN.TIME.F ST = max (the min-val of ST, min (the max-val of ST, the bias of ST + the gain of ST * sin (the current time / the frequency of ST)))
the output-1 of any COS.TIME.F CT = max (the min-val of CT, min (the max-val of CT, cos (the current time / the frequency of CT) / the gain of CT))
the output-1 of any EXP.RN.PDF E = max (the min-val of E, min (the max-val of E, - ln (random (0.0, 1.0)) * the mean of E))

TABLE 88

GENERIC SIMULATION FORMULAS for Cell Cycle

state variable : d / dt (the number-of-cells of any g1-phase P) = the progression-rate of the cycle-path connected at the G0 of P + the progression-rate of the cycle-path connected at the Di of P - the progression-rate of the cycle-path connected at the S of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any S-phase P) = the progression-rate of the cycle-path connected at the G1 of P - the progression-rate of the e-cycle-path connected at the G2 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any G2-phase P) = the progression-rate of the cycle-path connected at the S of P - the progression-rate of the e-cycle-path connected at the M of P - the progression-rate of the cycle-path connected at the Ap of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any M-phase P) = the progression-rate of the cycle-path connected at the G2 of P - the progression-rate of the cycle-path connected at the G0 of P - the progression-rate of the cycle-path connected at the G1 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any differ-stage P) = the progression-rate of the cycle-path connected at the G1 of P - the progression-rate of the e-cycle-path connected at the NG of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any apoptosis P) = the progression-rate of the cycle-path connected at the G2 of P, with initial value 0.0
state variable : d / dt (the number-of-cells of any cell-phase C) = the sum over each cycle- path Pi connected at an input of C of (the progression-rate of Pi) - the sum over each cycle-path Po connected at an output of C of (the progression-rate of Po) , with initial value 0.0
the progression-rate of any e-cycle-path P = the rate-constant of P * the tau-coeff of P * the number-of-cells of the cell-phase connected at the input of P

TABLE 89

	an user-menu-choice
Label	create-local
Applicable class	complex-bioentity
Condition	the item has a name
Action	start ET-CREATE-LOCAL-PROC (the item. this window)

ET-CREATE-LOCAL-PROC (EN: class complex-bioentity. W: class window)
  create a complex-bioentity LO by cloning EN:
  transfer LO to the WS of EN at (the item-x-position of EN +
    10. the item-y-position of EN - 30):
  for each bioentity CO upon the SW of LO do
    make CO transient and delete CO:
  for each bioentity-notes BN upon the SW of LO do
    make BN transient and delete BN:
   make LO permanent:

TABLE 90

	an user-menu-choice
Label	add-scaled-input
Applicable class	biopool
Condition	the scaled-input-model-box connected to the item does not exist
Action	start CREATE-SCALED-INPUT-MODEL-BOX-PROC (the item. this window)

CREATE-SCALED-INPUT-MODEL-BOX-PROC (BP: class biopool. Win: class window)
  create a scaled-input-model-box B by cloning MASTER-SCALED-
    INPUT-MODEL-BOX:
  transfer B to the workspace of BP at (left-edge(BP) -7. middle-y(BP)+56):
  make B permanent:
  show the subworkspace of B on Win with its center 340 units above the
   center of the screen:

TABLE 91

	an user-menu-choice
Label	create-sw
Applicable class	bioengine
Condition	not (there exists a subworkspace of the item) and the toggle-state of the item is hide
Action	start CREATE-SW-FOR-BIOENGINE (the item. this window)

CREATE-SW-FOR-BIOENGINE (E: class bioengine. win: class window)

  create a workspace box by cloning the SW of BIOENGINE-SW-MASTER:
  show box on win at the center of the screen:
  make box the subworkspace of E:
  make box permanent:

TABLE 92

an user-menu-choice

Label	add-top-post
Applicable class	biopool
Condition	none
Action	start CREATE-BIOPOOL-P-POST-PROC (the item, this window)

```
CREATE-BIOPOOL-P-POST-PROC (BP: class biopool, Win: class window)
 create a biopool-p-post P;
 if the biopool-p-post connected at the p02 of BP does not exist then
    next-label = "pp02";
    x = - 50; else
 if the biopool-p-post connected at the p04 of BP does not exist then
    next-label = "pp04";
    x = - 25; else
 if the biopool-p-post connected at the p06 of BP does not exist then
    next-label = "pp06";
    x = 0; else
 if the biopool-p-post connected at the p08 of BP does not exist then
    next-label = "pp08";
    x = 25; else
 if the biopool-p-post connected at the p10 of BP does not exist then
    next-label = "pp10";
    x = 50; else
 if the biopool-p-post connected at the p12 of BP does not exist then
    next-label = "pp12";
    x = 75; else
 go to part-2-routine;
 transfer P to the workspace of BP at (middle-x (BP) + x, middle-y (BP) + 40);
 change the text of the label of P to "[next-label]";
 call POOLPOST-NAME-IT-PROC (P, Win);
 make P permanent;
 return;
part-2-routine:
 if not (there exists a biopool-p-post connected at the p01 of BP) then
    next-label = "pp01";
    x = - 63; else
 if the biopool-p-post connected at the p03 of BP does not exist then
    next-label = "pp03";
    x = - 38; else
 if the biopool-p-post connected at the p05 of BP does not exist then
    next-label = "pp05";
    x = - 13; else
 if the biopool-p-post connected at the p07 of BP does not exist then
    next-label = "pp07";
    x = 12; else
 if the biopool-p-post connected at the p09 of BP does not exist then
    next-label = "pp09";
    x = 37; else
 if the biopool-p-post connected at the p11 of BP does not exist then
    next-label = "pp11";
    x = 62; else
 go to exit-routine;
 transfer P to the workspace of BP at (middle-x (BP) + x, middle-y (BP) + 60);
 change the text of the label of P to "[next-label]";
 call POOLPOST-NAME-IT-PROC (P, Win);
 make P permanent;
 return;
exit-routine:
   md = call uil-create-message-dialog ("md", "Change the label of the new post to the
     next consecutive number, such as pp13, place it in next  position, and connect it to
     the connection of the last raised post, and select name-it from its menu", win, the
     symbol none, 0, 0);
   transfer P to the workspace of BP at (middle-x (BP) + 100, middle-y (BP) + 100);
   make P permanent;
   if the biopool-p-post connected at the p01 of BP exists then
    move the biopool-p-post connected at the p01 of BP to (middle-x (BP)
      - 00, middle-y (BP) + 100);
```

TABLE 93

an user-menu-choice

Label	name-it
Applicable class	biopool-post
Condition	the item has no name
Action	start POOLPOST-NAME-IT-PROC (the item, this window)

POOLPOST-NAME-IT-PROC (post: class biopool-post, win: class window)
BR = the bioreservoir superior to the workspace of post;
length = length-of-text ("[the name of BR]");
base-name = omit-from-text ("[the name of BR]", length - 1, length);
post-name = insert-in-text ("[the label of post]", base-name, length - 1);
change the name of post to symbol ("[post-name]");
change the text of the label of post to "[post-name]";
change the face-color icon-color of post to green;
make post permanent;

TABLE 94

an user-menu-choice

Label	unname-it
Applicable class	biopool-post
Condition	the item has a name
Action	start POOLPOST-REMOVE-NAME-PROC (the item, this window)

POOLPOST-REMOVE-NAME-PROC (post: class biopool-post, W: class window)
change the text of the names of post to "none";
change the text of the ref-bioprocess of post to "none";
length = length-of-text (the label of post);
base-label = omit-from-text (the label of post, 1, length - 4);
change the text of the label of post to "[base-label]";
change the face-color icon-color of post to light-gray;
make post permanent;

TABLE 95

an user-menu-choice

Label	unname-it
Applicable class	biorole-post
Condition	the item has a name
Action	start ROLEPOST-REMOVE-NAME-PROC (the item, this window)

ROLEPOST-REMOVE-NAME-PROC (post: class biorole-post, win: class window)
if there exists a biopool connected to post then
BR = the bioreservoir superior to the workspace of the biopool connected to post;
if the biopool-post named by the name of post exists then
PP = the biopool-post named by the name of post;
change the text of the ref-bioprocess of PP to "none";
make PP permanent;
change the text of the ref-bioreservoir of post to "none";
change the text of the names of post to "none";
change the text of the label of post to "none";
make post permanent;

TABLE 96

an user-menu-choice

Label	set-refs
Applicable class	biorole-post
Condition	the item has a name
Action	start ROLEPOST-SET-REFS-PROC (the item, this window)

ROLEPOST-SET-REFS-PROC (post: class biorole-post, W: class window)
BP = the bioprocess superior to the workspace of post;
if there exists a biopool PO connected to post then
BR = the bioreservoir superior to the workspace of PO;
PP = the biopost named by the name of post;
if post is a bioreactant-post and PP is not a biopool-r-post then inform the operator that "You are trying ..." else
if post is a bioproduct-post and PP is not a biopool-p-post then inform the operator that "You are trying " else
conclude that the ref-bioprocess of PP = the name of BP;
make PP permanent;
conclude that the ref-bioreservoir of post = the name of BR;
make post permanent;

TABLE 97

```
initially    show the subworkspace of INITIALIZATION-PANEL at the center of the screen
```

TABLE 98

```
BASIS-MENU-HEAD-CALLBACK (button: class button, win: class window, dlg-or-wksp: class item)
  if the user-mode of win is developer then start gdm-init-and-show-menu-system (developer -menu-head, win) else
  if the user-mode of win is modeler then start gdm-init-and-show-menu-system (modeler -menu-head, win) else
  if the user-mode of win is general or the user-mode of win is navigation then start gdm-init-and-show-menu-system
        (general -menu-head, win) else
  if the user-mode of win is simulation then start gdm-init-and-show-menu-system (simulation-menu-head, win);
  return
```

TABLE 99

```
NAVIG-INIT-PROC (win: class window)
 inform the operator that "Initialization started";
 for BP = each bioprocess do
  allow other processing;
  if the subworkspace of BP exists then activate the subworkspace of BP;
 for BR = each bioreservoir do
  allow other processing;
  if the subworkspace of BR exists then activate the subworkspace of BR;
 call BR-INITIALIZATION-PROC ();
 inform the operator that "The bioReservoirs have been initialized";
 for BP = each bioprocess do
  if BP has a name then
     call BP-INITIALIZATION-PROC (BP);
     call DOWNSTREAM-CHAINING-PROC (BP);
     call UPSTREAM-CHAINING-PROC (BP);
 inform the operator that "The bioProcesses have been initialized and chaining is completed";
 for BM = each biomodel do
  call BM-INITIALIZATION-PROC (BM);
 inform the operator that "The bioModels have been initialized";
```

TABLE 100

```
BR-INITIALIZATION-PROC ()
 for BR = each bioreservoir do
  if BR has a name then
     allow other processing;
     conclude that the status of BR is available;
     SW = the subworkspace of BR;
     PO = the biopool upon SW;
     conclude that the label of PO = the label of BR;
     change the text of the master-bioreservoir of BR to"[the name of BR]";
     WS = the workspace of BR;
     if the bio-object-title upon WS exists then
        conclude that the compartment of BR = "[the label of the bio-object-title upon WS]";
     if the warnings of BR /= "" then change the flag-color icon-color of BR to yellow else change the flag-color icon-
        color of BR to the status-color icon-color of BR;
     if there exists a biopool-r-post connected to PO then
       for PRP = each biopool-r-post connected to PO do
        if PRP has a name then
           if there exists a bioreactant-post RP such that (RP has a name and the name of RP = the name of PRP) then
              DBP = the bioprocess superior to the workspace of RP;
              conclude that the ref-bioprocess of PRP = the name of DBP;
              conclude that DBP is a-down-bioprocess-of BR;
     if there exists a biopool-p-post connected to PO then
       for PPP = each biopool-p-post connected to PO do
        if PPP has a name then
           allow other processing;
           if there exists a bioproduct-post PP such that (PP has a name and the name of PP = the name of PPP) then
              UBP = the bioprocess superior to the workspace of PP;
              conclude that the ref-bioprocess of PPP = the name of UBP;
              conclude that UBP is an-up-bioprocess-of BR;
 make BR permanent;
 return
```

TABLE 101

```
BP-INITIALIZATION-PROC (BP: class bioprocess)

 SW: class kb-workspace  = the subworkspace of BP:
 WS: class kb-workspace = the workspace of BP:
 allow other processing:
 conclude that the status of BP is available:
 if there exists a bioengine E upon SW then
  change the type-color icon-color of BP to the type-color icon-color
    of the bioengine upon SW:
  conclude that the label of E = the label of BP:
  change the text of the master-bioprocess of BP to "[the name of BP]":
  if the bio-object-title upon WS exists then
     conclude that the compartment of BR = "[the label of the bio-object-title upon WS]":
  if the warnings of BP /= "" then change the flag-color icon-color of BP to yellow else
     change the flag-color icon-color of BP to the type-color icon-color of BP:
  if there exists a bioreactant upon SW then
    for R = each bioreactant upon SW do
     allow other processing:
     RP = the bioreactant-post connected to R:
     if RP has a name then
      if there exists a biopost PRP such that (PRP has a name and
        the name of PRP = the name of RP and PRP is not a biopool-
        r-post) then inform the operator that "A wrong connection
        has been establised.  A bioreactant-post named [the name
        of RP] has been given the same name as a biopost that is
        not a biopool-r-post" else
       if there exists a biopool-r-post PRP such
         that (PRP has a name and the name of PRP = the name of RP) then
       UBR = the bioreservoir superior to the workspace of PRP:
       conclude that the ref-bioreservoir of RP = the name of UBR:
       if there exists a bioprocess that is an-up-bioprocess-of UBR then
         for USBP = each bioprocess that is an-up-bioprocess-of UBR do
           conclude that USBP is an-upstream-bioprocess-of BP:
           if there exists a bioreservoir that is an-up-bioreservoir-of USBP then
             for USBR = each bioreservoir that is an-up-bioreservoir-of USBP do
               conclude that USBR is an-upstream-bioreservoir-of UBR:
  if there exists a bioproduct upon SW then
   for P = each bioproduct upon SW do
    allow other processing:
    PP = the bioproduct-post connected to P:
    if PP has a name then
     if there exists a biopost PPP such that (PPP has a name and
       the name of PPP = the name of PP and PPP is not a biopool-
       p-post) then inform the operator that "A wrong connection
       has been establised.  A bioproduct-post named [the name of
       PP] has been given the same name as a biopost that is not
       a biopool-p-post" else
      if there exists a biopool-p-post PPP such
        that (PPP has a name and the name of PPP = the name of PP) then
      DBR = the bioreservoir superior to the workspace of PPP:
      conclude that the ref-bioreservoir of PP = the name of DBR:
      if there exists a bioprocess that is a-down-bioprocess-of DBR then
        for DSBP = each bioprocess that is a-down-bioprocess-of
            DBR do
          allow other processing:
          conclude that DSBP is a-downstream-bioprocess-of BP:
          if there exists a bioreservoir that is a-down-bioreservoir-of
              DSBP then
            for DSBR = each bioreservoir that is a-down-bioreservoir-of DSBP do
              conclude that DSBR is a-downstream-bioreservoir-of DBR:
 make BP permanent.
 return:
```

TABLE 102

```
DOWNSTREAM-CHAINING-PROC (BP: class bioprocess)
if there exists a bioprocess that is a-downstream-bioprocess-of BP then
  for BP1 = each bioprocess that is a-downstream-bioprocess-of BP do
   allow other processing:
   if there exists a bioprocess that is a-downstream-bioprocess-of BP1 then
    for DBP = each bioprocess that is a-downstream-bioprocess-of BP1 do
     if DBP is not a-downstream-bioprocess-of BP then
       allow other processing:
       conclude that DBP is a-downstream-bioprocess-of BP:
       call DOWNSTREAM-CHAINING-PROC (DBP):
 if there exists a bioreservoir that is a-down-bioreservoir-of BP then
  for BR1 =each bioreservoir that is a-down-bioreservoir-of BP do
   allow other processing:
   if there exists a bioreservoir that is a-downstream-bioreservoir-of BR1 then
    for BR2 =each bioreservoir that is a-downstream-bioreservoir-of BR1 do
     if there exists a bioreservoir that is a-downstream-bioreservoir-of BR2 then
      for DBR = each bioreservoir that is a-downstream-bioreservoir-of BR2 do
       if DBR is not a-downstream-bioreservoir-of BR1 then
         allow other processing:
         conclude that DBR is a-downstream-bioreservoir-of BR1:
```

TABLE 103

```
UPSTREAM-CHAINING-PROC (BP: class bioprocess)
 if there exists a bioprocess that is an-upstream-bioprocess-of BP then
  for BP1 = each bioprocess that is an-upstream-bioprocess-of BP do
   allow other processing:
   if there exists a bioprocess that is an-upstream-bioprocess-of BP1 then
    for UBP = each bioprocess that is an-upstream-bioprocess-of BP1 do
     if UBP is not an-upstream-bioprocess-of BP then
       allow other processing:
       conclude that UBP is an-upstream-bioprocess-of BP:
       call UPSTREAM-CHAINING-PROC (UBP):
 if there exists a bioreservoir that is an-up-bioreservoir-of BP then
  for BR1 =each bioreservoir that is an-up-bioreservoir-of BP do
   allow other processing:
   if there exists a bioreservoir that is an-upstream-bioreservoir-of BR1 then
    for BR2 =each bioreservoir that is an-upstream-bioreservoir-of BR1 do
     if there exists a bioreservoir that is an-upstream-bioreservoir-of BR2 then
      for UBR = each bioreservoir that is an-upstream-bioreservoir-of BR2 do
       if UBR is not an-upstream-bioreservoir-of BR1 then
         allow other processing:
         conclude that UBR is an-upstream-bioreservoir-of BR1:
```

TABLE 104

```
INIT-BM-CONTAINED-CALLBACK (button: class uil-button, win: class window, wksp: class kb-workspace)
  inform the operator that "New biomodel names and containing relations will now be established":
  for each biomodel BM do:
    call BIOMODEL-INITIALIZATION-PROC (BM):
  inform the operator that "New biomodel names and containing relations are now completed"
  return
```

```
BIOMODEL-INITIALIZATION-PROC (BM: class biomodel)
  view = the SW of BM
  set the title upon view equal to "the label of BM - the title of view":
  for each biomodel BM1 upon view do:
    conclude that BM1 is BM-CONTAINED-IN BM:
    for each biomodel CBM that is the-container-of-bm BM do:
      conclude that BM1 is BM-CONTAINED-IN CBM:
    call BIOMODEL-INITIALIZATION-PROC (BM1):
  if the warnings of BM is not empty then
    change the flag-color of BM to yellow else change the flag-color of BM to gray:
  for each bioProcess BP upon view do:
    conclude that BP is BP-CONTAINED-IN BM:
    for each biomodel CBM that is the-container-of-bm BM do:
      conclude that BP is BP-CONTAINED-IN CBM:
  for each bioReservoir BR upon view do:
    conclude that BR is BR-CONTAINED-IN BM:
    for each biomodel CBM1 that is the-container-of-bm BM do
      conclude that BR is BR-CONTAINED-IN CBM:
  return
```

TABLE 105

```
ADD-QUERY-CALLBACK (button: class uil-button, win: class window, panel: class selection-panel)
 call ET-INITIALIZATION-PROC ();
 call INIT-QUERY-TABLES-PROC (win);
 return
```

TABLE 106

```
QUERY-INIT-CALLBACK (button: class button, win: class window, panel: class selection-panel)
 call NAVIG-INIT-PROC (win);
 call ET-INITIALIZATION-PROC ();
 call INIT-QUERY-TABLES-PROC (win);
 return
```

TABLE 107

```
ET-INITIALIZATION-PROC ()
n: integer = 0;
 for ET = each complex-bioentity do
  if ET has a name then
     n = 0;
     conclude that the id of ET = "[the name of ET]-0";
     for CE = each complex-bioentity that is of the class named by the class of ET do
      if the master-bioentity of CE = the name of ET then
        n = n + 1;
        conclude that the id of CE = "[the name of ET]-[n]";
 return ;
```

TABLE 108

```
INIT-QUERY-TABLES-PROC (win: class g2-window)
 activate the subworkspace of QUERY-ARRAYS-BIN;
 if there exists a query-array upon the subworkspace of QUERY-ARRAYS-BIN then
  for QA = each query-array upon the subworkspace of QUERY-ARRAYS-BIN do
   make QA transient;
   delete QA;
 if there exists a query-list upon the subworkspace of QUERY-LISTS-BIN then
  for QL = each query-list upon the subworkspace of QUERY-LISTS-BIN do
   delete QL;
 call FIND-BIOENTITY-DEFINITIONS-PROC ();
 call CREATE-QUERY-LISTS-PROC ();
 for QL = each query-list upon the subworkspace of QUERY-LISTS-BIN do
  CL = the ref-component-class of QL;
  if there exists a bioentity EC that is of the class named by CL then
   for EC = each bioentity that is of the class named by CL do
    call FIND-NAMED-BIOENTITIES-WITH-COMPONENT-PROC (EC, QL, win);
    conclude that the lenght of QL = the number of elements in QL;
 call COMPLETE-QUERY-TABLE-PROC (win);
 call CREATE-AUXILIARY-QUERY-ARRAYS-PROC (win);
 deactivate the subworkspace of QUERY-ARRAYS-BIN;
 activate the subworkspace of QUERY-ARRAYS-BIN;
 inform the operator that "The master query lists have been completed"
```

TABLE 109

```
FIND-BIOENTITY-DEFINITIONS-PROC ()
 for B = each uil-goto-workspace-button upon the subworkspace of BIOENTITIES do
  for B1 = each uil-goto-workspace-button upon the subworkspace of B do
   if there exists an object-definition upon the subworkspace of B1 then
    for OD = each object-definition upon the subworkspace of B1 do
     call CREATE-QUERY-ARRAY-PROC (OD)
   else
    for B2 = each uil-goto-workspace-button upon the subworkspace of B1 do
     if there exists an object-definition upon the subworkspace of B2 then
      for OD = each object-definition upon the subworkspace of B2 do
       allow other processing;
       call CREATE-QUERY-ARRAY-PROC (OD)
     else
      for B3 = each uil-goto-workspace-button upon the subworkspace of B2 do
       if there exists an object-definition upon the subworkspace of B3 then
        for OD = each object-definition upon the subworkspace of B3 do
         call CREATE-QUERY-ARRAY-PROC (OD)
```

TABLE 110

```
CREATE-QUERY-ARRAY-PROC (MC: class object-definition)
 SC = symbol (the text of the superior-class of MC):
 SW = the subworkspace of query-arrays-bin
   case (SC) of
     protein-site :
       create a query-array MA:
       transfer MA to SW at (-160. -265):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "protein-site":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     protein-modified-group :
       create a query-array MA:
       transfer MA to SW at (-160. -265):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "protein-modified-group":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array").
     protein-motif:
       create a query-array MA:
       transfer MA to SW at (-160. -140):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "protein-motif":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     protein-domain:
       create a query-array MA:
       transfer MA to SW at (-160. -10):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "protein-domain":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     protein:
       create a query-array MA:
       transfer MA to SW at (-160. 120):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "protein":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     heter.mol.complex:
       create a query-array MA:
       transfer MA to SW at (-160. 245):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "heter.mol.complex":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     dna-component :
       create a query-array MA:
       transfer MA to SW at (-160. 370):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "dna-component":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     nucleic-acid:
       create a query-array MA:
       transfer MA to SW at (-160. 370):
       CL = the text of the class-name of MC:
       change the text of the major-class of MA to "nucleic-acid":
       change the text of the ref-component-class of MA to "[CL]":
       change the name of MA to symbol ("[CL]-query-array"):
     otherwise.
       allow other processing:
```

TABLE 111

```
CREATE-QUERY-LISTS-PROC ()

SW = the subworkspace of QUERY-ARRAYS-BIN :
 for QA = each query-array upon SW do
  allow other processing:
  QAN = the name of QA:
  create a query-list QL:
  transfer QL to SW at (middle-x(QA), middle-y (QA)):
  change the text of the ref-array of QL to "[QAN]":
  change the text of the ref-component-class of QL to "[the ref-
    component-class of QA]":
```

TABLE 112

```
FIND-NAMED-BIOENTITIES-WITH-COMPONENT-PROC (EC: class bioentity ,
QL: class query-list, win: class window)

 if EC has a name then
  call SEEK-BIOENTITY-COPIES-PROC (EC, QL, win)
 else
   allow other processing:
   SECN = call FIND-NAMED-BIOENTITY (EC, win):
   if SECN is not none then
     allow other processing:
     SEC = the bioentity named by SECN:
     call SEEK-BIOENTITY-COPIES-PROC (SEC, QL, win):
 return:
```

TABLE 113

```
FIND-NAMED-BIOENTITY (E: class bioentity, win: class window) = (symbol )

 SE1 = the object superior to the workspace of E:
 if SE1 is not a bioentity then return the symbol none else
 if SE1 has a name then return the name of SE1 else
   SE2 = the object superior to the workspace of SE1:
   if SE2 is not a bioentity then return the symbol none else
   if SE2 has a name then return the name of SE2 else
     SE3 = the object superior to the workspace of SE2:
     if SE3 is not a bioentity then return the symbol none else
     if SE3 has a name then return the name of SE3 else
       SE4 = the object superior to the workspace of SE3:
       if SE4 is not a bioentity then return the symbol none else
       if SE4 has a name then return the name of SE4 else
         SE5 = the object superior to the workspace of SE4:
         if SE5 is not a bioentity then return the symbol none else
         if SE5 has a name then return the name of SE5 else
           SE6 = the object superior to the workspace of SE5:
           if SE6 is not a bioentity then return the symbol none else
           if SE6 has a name then return the name of SE6:
```

TABLE 115

COMPLETE-QUERY-TABLE-PROC (win: class g2-window)

```
ASW = the subworkspace of QUERY-ARRAYS-BIN:
LSW = the subworkspace of QUERY-LISTS-BIN:
if there exists a query-list upon LSW then
  for QL = each query-list upon LSW do
    if the lenght of QL > 0 then
      if there exists a query-array QA upon ASW such that (QA is the same
          object as the query-array named by the ref-array of QL) then
        allow other processing:
        change the array-length of QA to the lenght of QL:
        for S = each symbol in QL do
          IV = the text of the initial-values of QA:
          change the text of the initial-values of QA to "[IV], [S]":
        IV = omit-from-text (the text of the initial-values of QA, 1, 6):
        change the text of the initial-values of QA to "[IV]":
        make QA permanent:
```

TABLE 116

	a sup-ct-array
Names	A-HELIX-MOTIF-QUERY-ARRAY
Array length	28
Element type	symbol
Initial values	a.helix.domain.mc, gh.ma, prolactin.ma, f.somatolactin.ma, m.epo.ma, il2.ma, m.il6.ma, m.g.csf.ma, c.mgf.ma, il6.ma, g.csf.ma, m.lif.ma, osm.ma, r.cntf.ma, lif.ma, m.il4.ma, m.il3.ma, m.gm.csf.ma, m.il5.ma, il4.ma, il5.ma, il3.ma, gm.csf.ma, m.il7.ma, m.il9.ma, il7.ma, il9.ma, a.helix.motif.mc
Ref component class	a-helix-motif
Major class	protein-motif

	a query-list
Names	A-HELIX-MOTIF-QUERY-LIST
Element type	symbol
Allow duplicate elements?	no
Ref component class	a-helix-motif
Ref array	a-helix-motif-query-array
Lenght	28

TABLE 117

CREATE-AUXILIARY-QUERY-ARRAYS-PROC (win: class window)

```
SW = the subworkspace of QUERY-ARRAYS-BIN
create a query-array MA:
transfer MA to SW at (-160, -390):
change the name of MA to the symbol no-group-selection-query-array:
make MA permanent:
create a query-array MA:
transfer MA to SW at (-160, -390):
change the name of MA to the symbol no-motif-selection-query-array:
make MA permanent:
create a query-array MA:
transfer MA to SW at (-160, -390):
change the name of MA to the symbol no-domain-selection-query-array:
make MA permanent:
create a query-array MA:
transfer MA to SW at (-160, -390):
change the name of MA to the symbol no-subunit-selection-query-array:
make MA permanent:
```

TABLE 118

```
INIT-ICONS-CALLBACK (button: class button, win: class window, panel : class
selection-panel )

 call INIT-BR-ICONS-PROC ();
 call INIT-BP-ICONS-PROC ();
 return
```

```
INIT-BR-ICONS-PROC ()
 for BR = each bioreservoir do
  if the subworkspace of BR exists then
    allow other processing;
    WS = the subworkspace of BR;
    P = the biopool upon WS;
    for GL = each e-graph-link connected to P do
      change the core stripe-color of GL to transparent;
    if there exists a e-graph-tracer upon WS then
      SG = the e-scaled-br-graph-tracer connected to P;
      move SG to (left-edge(P) - 8, middle-y(P) + 11);
      AG = the e-absolute-br-graph-tracer connected to P;
      move AG to (left-edge(P) - 8, middle-y(P) - 10);
      CH = the e-br-chaining-tracer connected to P;
      move CH to (right-edge(P) + 7, middle-y(P));
    change the size of WS to minimum;
    make WS permanent;
```

```
INIT-BP-ICONS-PROC ()
 for BP = each bioprocess do
  if the subworkspace of BP exists then
    allow other processing;
    WS = the subworkspace of BP;
    E = the bioengine upon WS;
    for GL = each e-graph-link connected to E do
      change the core stripe-color of GL to transparent;
    ST = the e-scaled-bp-graph-tracer connected to E;
    move ST to (left-edge(E) - 9, middle-y(E) + 11);
    CH = the e-bp-chaining-tracer connected to E;
    move CH to (left-edge(E) - 9, middle-y(E) - 11);
    UP = the e-up-path-tracer connected to E;
    move UP to (right-edge(E) + 9, middle-y(E) + 11);
    DN = the e-down-path-tracer connected to E;
    move DN to (right-edge(E) + 9, middle-y(E) - 11);
    if there exists a bioreactant-post upon WS then
      for RP = each bioreactant-post upon WS do
        R = the bioreactant connected to RP;
        move RP to (left-edge(R) - 3, middle-y(R));
    if there exists a bioproduct-post upon WS then
      for PP = each bioproduct-post upon WS do
        P = the bioproduct connected to PP;
        move PP to (right-edge(P) + 4, middle-y(P));
      change the size of WS to minimum;
      make WS permanent;
```

TABLE 118b

```
ALL-INIT-CALLBACK (button: class button, win: class window, panel: class
selection-panel)
 call NAVIG-INIT-PROC (win);
 call SIMUL-INIT-PROC (win);
 call INIT-BR-ICONS-PROC ();
 call INIT-BP-ICONS-PROC ();
 return
```

TABLE 119

```
for any biomodel BM
whenever BM is moved by the user then
start BIOMODEL-WS-CHECK-PROC (BM)
```

```
BIOMODEL-WS-CHECK-PROC (BM: class biomodel)
  WS = the workspace of BM;
  if there is a title upon WS and BM has a SW and there is a title upon the
  SW of BM then:
    T = the title upon the SW of BM;
    new-label = "<the label of BM>-<the label of the title upon WS>";
    if new-label /= the label of T then:
       conclude that the label of T = new-label;
       call REMOVE-BIOMODEL-CONTAINING-PROC (BM);
       call BIOMODEL-INITIALIZATION-PROC (BM);
```

```
REMOVE-BIOMODEL-CONTAINING-PROC (BM: class biomodel)
  WS = the workspace of BM;
  for each biomodel BM1 that is bm-contained-in BM do
     for each biomodel CBM that is the-container-of-bm BM do
        conclude that BM1 is not BM-CONTAINED-IN CBM;
  for each bioprocess BP that is bp-contained-in BM do
     for each biomodel CBM that is the-container-of-bm BM do
        conclude that BP is not BP-CONTAINED-IN CBM;
  for each bioreservoir BR that is br-contained-in BM do
     for each biomodel CBM that is the-container-of-bm BM do
        conclude that BR is not BR-CONTAINED-IN CBM;
  for each biomodel CBM that is the-container-of-bm BM do
     conclude that BM is not BM-CONTAINED-IN CBM;
  if WS has a superior item S and S is a biomodel then:
     BM2 = the biomodel superior to WS;
     conclude that BM is BM-CONTAINED-IN BM2;
     for each biomodel CBM that is the-container-of-bm BM2 do
        conclude that BM is BM-CONTAINED-IN CBM;
  return
```

TABLE 120

an user-menu-choice

Label	details
Applicable class	bioprocess
Condition	the subworkspace of the item exists
Action	start DETAILS-BP-PROC (the item. this window)

```
DETAILS-BP-PROC (BP: class bioprocess. WIN: class window)
 if the subworkspace of BP exists then
  SW = the subworkspace of BP:
  if the toggle-state of BP is hide then:
    if the status of bP is available then
      conclude that the status of bP is selected:
      activate SW:
      change the status-color of BP to brown:
      conclude that the toggle-state of BP is show:
    E = the bioengine upon SW :
    move the scaled-bp-graph-tracer connected to E to (left-edge(E) - 9.
    middle-y(E) + 11):
    move the bp-chaininig-tracer connected to E to (right-edge(E) - 9.
    middle-y(E) - 10):
    move the up-path-tracer connected to E to (right-edge(E) + 9.
    middle-y(E) + 11):
    move the down-path-tracer connected to E to (right-edge(E) + 9.
      middle-y(E) - 10):
    for each bioreactant R upon view do
      move the bioreactant-post connected to R to (left-edge(R) - 3.
        middle-y(R)):
    for P = each bioproduct upon view do
      move the bioproduct-post connected to P to (right-edge(P) + 4.
      middle-y(P)):
    change the size of SW to minimum:
    show SW on Win at the center of the screen:
  else if the toggle-state of BP is show then:
    conclude that the toggle-state of BP is hide:
    change the status-color of BP to slate-blue:
    if the status of bP is selected then
      conclude that the status of bP is available:
    deactivate SW :
    hide SW on Win
 else if the master-bioprocess of BP has a value then
   call BP-MASTER-DETAILS-PROC (BP. win)
 else return
```

```
BP-MASTER-DETAILS-PROC (BP: class bioprocess. win: class window)
 MA = the bioprocess named by the master-bioprocess of BP:
 if the toggle-state of BP is hide then
   conclude that the toggle-state of BP is show:
   if the subworkspace of the item named by the master-bioprocess of
     BP exists then
      SW = the subworkspace of MA:
      change the size of SW to minimum:
      show SW at full scale with its center at the center of the screen:
      conclude that the toggle-state of MA is show:
 else
 if the toggle-state of BP is show then
    conclude that the toggle-state of BP is hide:
    conclude that the toggle-state of MA is hide:
    hide the subworkspace of MA on win:
```

TABLE 121

an user-menu-choice

Label	details
Applicable class	bioreservoir
Condition	none
Action	start DETAILS-BR-PROC (the item, this window)

```
DETAILS-BR-PROC (BR: class bioreservoir, win: class window)
if the subworkspace of Br exists then
 SW = the subworkspace of BR:
 if the toggle-state of BR is hide then
    change the status-color icon-color of BR to brown:
    conclude that the toggle-state of BR is show:
    CL = the br-chainnig-tracer connected to the biopool upon SW:
    SG = the scaled-br-graph-tracer connected to the biopool upon SW:
    AG = the absolute-br-graph-tracer connected to the biopool upon SW:
    move CL to (right-edge(P) + 6, middle-y(P)):
    move SG to (left-edge(P) - 8, middle-y(P) + 11):
    move AG to (left-edge(P) - 8, middle-y(P) - 10):
    change the size of SW to minimum:
    show the subworkspace of BR on win at full scale with its bottom
      left corner 200 units to the right of the center of the screen:
 else
    conclude that the toggle-state of BR is hide:
    change the status-color icon-color of BR to medium-aquamarine:
    hide the subworkspace of BR on win:
else if the master-bioprocess of BP has a value then
  call BR-MASTER-DETAILS-PROC (BP, win:
else return
```

```
BR-MASTER-DETAILS-PROC (BR: class bioreservoir, win: class window)
 MA = the bioreservoir named by the master-bioreservoir of BR:
 if the toggle-state of BR is hide then
   conclude that the toggle-state of BR is show:
   if the subworkspace of the item named by the master-bioreservoir of BR exists then
     SW = the subworkspace of MA:
     change the size of SW to minimum:
     show SW at full scale with its center at the center of the screen:
     conclude that the toggle-state of MA is show:
     if the text of the user-mode of win /= "simulation" then
       change the status-color icon-color of MA to brown:
 else
    conclude that the toggle-state of BR is hide:
    conclude that the toggle-state of MA is hide:
    hide the subworkspace of MA on win:
     if the text of the user-mode of win /= "simulation" then
       change the status-color icon-color of MA to slate-blue:
```

TABLE 122

an user-menu-choice

Label	details
Applicable class	complex-bioentity
Condition	none
Action	start ET-DETAILS-PROC (the item, this window)

```
ET-DETAILS-PROC (EN: class complex-bioentity, W: class window)
  if the subworkspace of EN exists then view = the subworkspace of EN
  if the toggle-state of EN is hide then
      conclude that the toggle-state of EN is show:
      show view at the center of the screen :
   if the master-bioentity of the item has a value then
      if the bioentity named by the master-bioentity of EN has a SW then
         master-view = the subworkspace of the bioentity named by the master-bioentity of EN:
       change the size of master-view to minimum:
       show master-view at the center of the screen:
  else
    if the toggle-state of EN is show then
      conclude that the toggle-state of EN is hide:
      hide view on W:
      hide master-view on W
  return:
```

TABLE 123

an user-menu-choice

Label	show-br
Applicable class	biorole-post
Condition	the item has a name
Action	start ROLEPOST-SHOW-BR-PROC (the item, this window)

```
ROLEPOST-SHOW-BR-PROC (post: class biorole-post, W: class window)
 BP = the bioprocess superior to the workspace of post:
 if post has a name then BR = the bioreservoir named by the ref-bioreservoir of post
 else
 if there exists a bioreservoir ABR such that (ABR has no name and
     the master-bioreservoir of ABR has a current value and the
     master-bioreservoir of ABR = the ref-bioreservoir of post) then BR = ABR:
 change the size of the subworkspace of BP to minimum:
 case (the toggle-state of post ) of
  show:
    change the color-pattern of post so that light-color is dim-gray , dark-color is light-gray :
    conclude that the toggle-state of post is hide:
    case (the class of post) of
     bioreactant-post :
       show the subworkspace of BR on W at full scale with its
         center 400 units to the left of the center of the screen:
       change the color-pattern of post so that white-color = the symbol black:
     bioproduct-post:
       show the subworkspace of BR on W at full scale with its
         center 400 units to the right of the center of the screen:
  hide:
    change the color-pattern of post so that light-color is light-gray, dark-color is dim-gray :
    conclude that the toggle-state of post is show:
    hide the subworkspace of BR on W:
    if the class of post is bioreactant-post then
     change the color-pattern of post so that white-color = the symbol white:
```

TABLE 124

an user-menu-choice

Label	show-bp
Applicable class	biopool-post
Condition	the item has a name and the ref-bioprocess of the item has a value
Action	start POOLPOST-SHOW-BP-PROC (the item, this window)

```
POOLPOST-SHOW-BP-PROC (post: class biopool-post, win: class window)
 BP = the bioprocess named by the ref-bioprocess of post
 change the size of the subworkspace of BP to minimum:
 case (the toggle-state of post ) of
  show :
    change the color-pattern of post so that light-color is dim-gray ,
     dark-color is light-gray , white-color = the symbol black:
    conclude that the toggle-state of post is hide:
    case (the class of post) of
     biopool-p-post:
       show the subworkspace of BP on W at full scale with its
          center 280 units above the center of the screen:
     biopool-r-post:
       show the subworkspace of BP on W at full scale with its
          center 280 units below the center of the screen:
  hide:
    change the color-pattern of post so that light-color is light-gray ,
     dark-color is dim-gray , white-color = the symbol white:
    conclude that the toggle-state of post is show:
    hide the subworkspace of BP on W:
```

TABLE 125

an user-menu-choice	
Label	bioentity
Applicable class	bioreservoir
Condition	the ref-bioentity of the item has a value
Action	change the size of the subworkspace SW of the bioentity named by the ref-bioentity of the item to minimum and show SW on win at the center of the screen:

an user-menu-choice	
Label	bioentity
Applicable class	biopool
Condition	the ref-bioentity of the bioreservoir superior to the workspace of the item has a value
Action	change the size of the subworkspace SW of the ref-bioreservoir superior to the workspace of the item to minimum and show SW on this window at the center of the screen

an user-menu-choice	
Label	bioentity
Applicable class	biorole-object
Condition	the ref-bioentity of the bioreservoir superior to the biopool connected to the item has a value
Action	change the size of the subworkspace SW of the ref-bioentity of the bioreservoir superior to the biopool connected to the item to minimum and show SW on this window at the center of the screen

TABLE 126

an user-menu-choice	
Names	none
Label	show-references
Applicable class	bioreservoir
Condition	none
Action	start SHOW-REFERENCES-OF-ARRAY-PROC (the item, this window)

```
SHOW-REFERENCES-OF-ARRAY-PROC (BR: class bioreservoir, win: class window)
  array = the references-array that gives the references of BR:
  BIN  = the subworkspace of REFERENCES-BIN:
  x = 0:
  y = 0:
  if the array-length of array /= 0 then
    create a workspace WS by cloning the subworkspace of REFERENCE-WS:
    for ref-button = each go-to-subws-button upon BIN  do
      SW = the subworkspace of ref-button:
      for R = each cabe-reference upon SW do
        for T = each text in array do
          if T /= "" and the short-ref of R = T then
            create a cabe-reference R1 by cloning R:
            transfer R1 to WS at (x, y):
            y = y - 50:
            if y < -900 then
              x = x + 300:
              y = 0:
  change the size of WS to minimum:
  show WS on win at the center of the screen:
```

TABLE 127

```
CREATE-GENERAL-STRUCTURE-SELECTION-PANEL-CALLBACK (menu: class menu-option.
win: class window)
 md = call create-message-dialog ("md", "Processing . . . . .  ", the symbol extra-large, win, 0, 0):
 panel = call clone-panel (MASTER-GENERAL-STRUCTURE-QUERY-PANEL):
 SW = the subworkspace of panel:
 call SET-RADIO-BOX-ID-PROC ("general-structure-panel",
       "GENERAL-STRUCTURE-QUERY-PANEL-radio-box", SW, win):
 if there exists a uil-medium-text-pushbutton de upon SW such that (the id of de = "de") then
   change the text of the callback of de to "DELETE-QUERY-PANEL-CALLBACK":
 show SW on win at the center of the screen:
 delete md:
 return:
```

TABLE 128

```
SET-RADIO-BOX-ID-PROC (r-id: text, rb-id: text, SW: class workspace, win: class window)
 if there exists a radio-button r upon SW such that (the id of r = r-id) then
    rb = the radio-box that is the-group-master-of r:
    urb-id = call unique-id ("[rb-id]"):
    conclude that the id of rb = "[urb-id]":
 return:
```

```
GET-RADIO-BOX-ID-PROC (r-id: text, SW: class workspace, win: class window) = (value )
 if there exists a radio-button r upon SW such that (the id of r = r-id) then
   rb = the adio-box that is the-group-master-of r:
 return "[the id of rb]":
```

TABLE 129

```
SELECT-GENERAL-STRUCTURE-QUERY-PANEL-CALLBACK (button: class button.
win: class window, dlg: class query-panel)

 SW = the subworkspace of dlg:
 md = call create-message-dialog ("md", "Processing . . . . .  ", the symbol extra-large, win, 0, 0):
 rb-id = call GET-RADIO-BOX-ID-PROC ("general-structure-panel", SW, win):
 type = call get-radio-box-value (dlg, rb-id):
 case (type) of
  subunits:
   MP = MASTER-SUBUNITS-QUERY-PANEL:
  domains:
   MP = MASTER-DOMAINS-QUERY-PANEL:
  motifs:
   MP = MASTER-MOTIFS-QUERY-PANEL:
  groups:
   MP = MASTER-GROUPS-QUERY-PANEL:
  subunits-subunits:
   MP = MASTER-SUBUNITS-SUBUNITS-QUERY-PANEL:
  domains-domains:
   MP = MASTER-DOMAINS-DOMAINS-QUERY-PANEL:
  domains-motifs:
   MP = MASTER-DOMAINS-MOTIFS-QUERY-PANEL:
  domains-groups:
   MP = MASTER-DOMAINS-GROUPS-QUERY-PANEL:
  motifs-motifs:
   MP = MASTER-MOTIFS-MOTIFS-QUERY-PANEL:
  motifs-groups:
   MP = MASTER-MOTIFS-GROUPS-QUERY-PANEL:
  groups-groups:
   MP = MASTER-GROUPS-GROUPS-QUERY-PANEL:
 panel = call clone-panel (MP):
 call CREATE-GENERAL-QUERY-PANEL-PROC (type, panel, win):
 show the subworkspace of dlg with its bottom right corner at the bottom right corner of the screen:
 show the subworkspace of panel at the center of the screen:
 delete md:
 return:.
```

TABLE 130

```
CREATE-GENERAL-QUERY-PANEL-PROC (type: symbol, panel: class query-panel,
win : class window)

 SW = the subworkspace of panel:
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  call SET-RADIO-BOX-ID-PROC ("subunits", "3-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  call SET-RADIO-BOX-ID-PROC ("domains", "4-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  call SET-RADIO-BOX-ID-PROC ("motifs", "5-BR-[type]-QUERY- radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  call SET-RADIO-BOX-ID-PROC ("groups", "6-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  call SET-RADIO-BOX-ID-PROC ("subunits", "3-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  call SET-RADIO-BOX-ID-PROC ("domains", "4-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  call SET-RADIO-BOX-ID-PROC ("motifs", "5-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  call SET-RADIO-BOX-ID-PROC ("groups", "6-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
  change the text of the callback of de to "DELETE-QUERY-PANEL-CALLBACK":
 show SW on win at the center of the screen:
 return:
```

TABLE 131

```
QUERY-GENERAL-STRUCTURE-CALLBACK (button: class button, win : class window,
panel: class query-panel)
    md = call uil-create-message-dialog ("md", "Processing . . . . .      ", the symbol large,
      win, 0, 0):
    ETL, TX2 = call QUERY-GENERAL-STRUCTURE-PROC (panel, win):
    create a free-text FT2:
    change the text of FT2 to "[TX2]":
    conclude that the scroll-title of ETL = TX2:
    conclude that FT2 is THE-TEST-ITEM-LIST-OF ETL:
    call CREATE-GENERAL-QUERY-OUTPUT-PANEL-PROC (ETL, FT2, panel, win):
 delete md:
 return:
```

TABLE 132

```
QUERY-GENERAL-STRUCTURE-PROC (panel: class query-panel, win : class window) =
(class scroll-text-list, text)

 PN = the name of panel:
 SW = the subworkspace of panel:
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  type1, PSA = call FIND-SPECIFIC-ARRAY-PROC ("subunits", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  type2, PSA-2 = call FIND-SPECIFIC-ARRAY-PROC ("subunits-2", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  type3, PDA = call FIND-SPECIFIC-ARRAY-PROC ("domains", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  type4, PDA-2 = call FIND-SPECIFIC-ARRAY-PROC ("domains-2", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  type5, PMA = call FIND-SPECIFIC-ARRAY-PROC ("motifs", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  type6, PMA-2 = call FIND-SPECIFIC-ARRAY-PROC ("motifs-2", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  type7, PGA = call FIND-SPECIFIC-ARRAY-PROC ("groups", panel, win):
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  type8, PGA-2 = call FIND-SPECIFIC-ARRAY-PROC ("groups-2", panel, win):
```

continue next page

TABLE 132 cont.

```
if PSA exists then
  if PSA-2 exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type1-type2-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol subunit. the symbol
        subunit. type1. type2. PSA. PSA-2. panel. win);
  else
    TX2 = capitalize-words ("proteins with structures defined in this module that contain [type1]");
    for T1 = each symbol in PSA do
      insert T1 at the end of the text list ETL;
else
if PDA exists then
  if PDA-2 exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type3-type4-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol domain. the symbol
        domain. type3. type4. PDA. PDA-2. panel. win)
  else
  if PMA exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type3-type5-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol domain. the symbol motif.
        type3. type5. PDA. PMA. panel. win)
  else
  if PGA exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type3-type7-ETL") then
      go to final-routine
    else
      ETL. TX2 = call C-MERGE-TWO-STRUCTURE-LISTS-PROC
        (the symbol domain. the symbol group. type3. type7. PDA. PGA. panel. win)
  else
    TX2 = capitalize-words ("proteins with structures defined in this module that contain [type3]");
    for T1 = each symbol in PDA do
      insert T1 at the end of the text list ETL;
else
if PMA exists then
  if PMA-2 exists then
    if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-type5-type6-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol motif. the symbol motif.
        type5. type6. PMA. PMA-2. panel. win)
  else
  if PGA exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type5-type7-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol motif. the symbol group.
        type5. type7. PMA. PGA. panel. win)
  else
    TX2 = capitalize-words ("proteins with structures defined in this module that contain [type5]");
    for T1 = each symbol in PMA do
      insert T1 at the end of the text list ETL;
else
if PGA exists then
  if PGA-2 exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-type7-type8-ETL") then
      go to final-routine
    else
      ETL. TX2 = call MERGE-TWO-STRUCTURE-LISTS-PROC (the symbol group. the symbol group.
        type7. type8. PGA. PGA-2. panel. win)
  else
    TX2 = capitalize-words ("proteins with structures defined in this module that contain [type7]");
    for T1 = each symbol in PGA do
      insert T1 at the end of the text list ETL;
return ETL. TX2;

final-routine:
  TX2 = the scroll-title of ETL;
  return ETL. TX2;
```

TABLE 133

```
FIND-SPECIFIC-ARRAY-PROC (r-id: text, panel: class query-panel, win : class window) =
(symbol, class query-array)
 SW = the subworkspace of panel:
 rb-id = call C-GET-UIL-RADIO-BOX-ID-PROC (r-id, SW, win);
 type = call uil-get-radio-box-value (panel, rb-id);
 array-name = symbol("[type]-QUERY-ARRAY");
 if the c-query-array named by array-name exists then QA = the c-query-array named by array-name
 else inform the operator that "[array-name] does not exist";
 move QA to (middle-x (QA) + 50, middle-y (QA));
 return type, QA;
```

TABLE 134

```
MERGE-TWO-STRUCTURE-LISTS-PROC (tag1: symbol, tag2: symbol, type4: symbol,
type5: symbol, QA1: class query-array, QA2: class query-array, panel: class query-panel,
win : class window) = (class scroll-text-list, text)
 PN = the name of panel:
 if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type4]-[type5]-ETL") then
  go to exit-routine
 else
   create a scroll-text-list ETL;
   conclude that the label of ETL = "[PN]-[type4]-[type5]-ETL";
   conclude that ETL is a-list-associated-to-dlg panel;
   transfer ETL to the subworkspace of safe-home-for-lists at (50, -245);
   TX2 = the scroll-title of ETL;
   if the array-length of QA1 = 0 then
     if the array-length of QA2 = 0 then
       if type5 = symbol ("no-[tag2]-selection") and type4 = symbol ("no-[tag1]-selection") then
         TX2 = capitalize-words ("no selection has been made")
       else
         if type5 = symbol ("no-[tag2]-selection") then
           TX2 = capitalize-words ("there are no structures defined in this module that contain [type4]")
         else
           if type4 = symbol ("no-[tag1]-selection") then
             TX2 = capitalize-words ("there are no structures defined in this module that contain [type5]")
           else
            TX2 = capitalize-words ("there are no structures defined in this module that contain [type4] or [type5]");
     else
       if type4 = symbol ("no-[tag1]-selection") then
        TX2 = capitalize-words ("proteins with structures defined in this module that contain [type5]")
       else
        TX2 = capitalize-words ("proteins with structures defined in this module that contain [type5]. None was
            found also containing [type4]");
        for T2 = each symbol in QA2 do
         insert "[T2]" at the end of the text list ETL;
   else
    if the array-length of QA2 = 0 then
       if type5 = symbol ("no-[tag2]-selection") then
        TX2 = capitalize-words ("proteins with structures defined in this module that contain [type4]")
       else
        TX2 = capitalize-words ("proteins with structures defined in this module that contain [type4]. None was
            found also containing [type5]");
       for T1 = each symbol in QA1 do
         insert "[T1]" at the end of the text list ETL;
    else
        TX2 = capitalize-words ("proteins with structures defined in this module that contain [type4] and [type5]");
        for T1 = each symbol in QA1 do
         for T2 = each symbol in QA2 do
          if T1 = T2 then
           insert "[T1]" at the end of the text list ETL;
 return ETL, TX2;
exit-routine:
  TX2 = the scroll-title of ETL;
  return ETL, TX2;
```

TABLE 135

```
CREATE-GENERAL-QUERY-OUTPUT-PANEL-PROC (ETL: class scroll-text-list. FT: class free-text.
dlg: class query-panel. win : class window) = (class query-panel)

 if there exists a query-panel panel such that (the text of the id of panel =
      "[the label of ETL]-QUERY-OUTPUT-PANEL.") then
   go to final-routine
 else
   create a query-panel panel by cloning MASTER-MOLECULAR-QUERY-OUTPUT-PANEL:
   transfer panel to the subworkspace of UIL-dialog-bin at (0.0):
   SW = the subworkspace of panel:
   change the text of the id of panel to "[the label of ETL]-QUERY-OUTPUT-PANEL":
   transfer FT to SW at (0. 90):
   make panel permanent:
   sa = call create-scroll-area-from-list ("query-general-scroll". 5. ETL):
   conclude that the allow-unselect-on-selected-message of sa is true:
   conclude that the allow-multiple-simultaneous-selections of sa is true:
   call add-obj-to-dialog (panel. sa. 0. -30):
   change the text of the uil-message-selection-method of sa to "query-message-selection":
   change the text of the uil-message-unselection-method of sa to query-message-unselection":
   change the text of the ordering-method of sa to "alphabetic":
   de = call create-pushbutton ("de". the symbol DELETE-QUERY-PANEL-CALLBACK.
        "DELETE". panel. 0. -145):
   go to final-routine:
final-routine :
 change the size of the subworkspace of panel to minimum:
 show the subworkspace of dlg with its top right corner 40 units below the top right corner of the screen:
 show the subworkspace of panel on win with its center 300 units to the left of the center of the screen:
 return panel:
```

TABLE 136

```
QUERY-MESSAGE-SELECTION-PROC (MO: class message-object. SA: class scroll-area.
win: class window)
BON = symbol (the text of MO):
BO = the bio-object named by BON:
 change the size of the subworkspace of BO to minimum:
 show the subworkspace of BO on win with its center 300 units to the
   right of the center of the screen :
```

```
QUERY-MESSAGE-UNSELECTION-PROC (MO: class message-object. SA: class scroll-area.
win: class window)
 BON = symbol (the text of MO):
 hide the subworkspace of the bio-object named by BON on win:
```

TABLE 137

```
DELETE-QUERY-PANEL-CALLBACK (button: class button. win : class window. panel: class dialog)
 SW = the subworkspace of DIALOG-BIN:
 md = call create-message-dialog ("md". the symbol DELETE-MD-CALLBACK. "Processing . . . . . ".
      the symbol extra-large. win. 0. 0):
 if there exists a scroll-text-list that is a-list-associated-to-dlg panel then
  for L = each scroll-text-list that is a-list-associated-to-dlg panel do
   delete L:
 call delete-dialog (panel):
 if there exists a message-dialog ME upon SW then
 for md = each message-dialog upon SW do
  delete md:
```

TABLE 138

an user-menu-choice

Label	query-panels
Applicable class	bioreservoir
Condition	the subworkspace of the item exists
Action	start CREATE-BR-RELATED-SELECTION-PANEL-CALLBACK (the item, this window)

an user-menu-choice

Label	show
Applicable class	query-tracer
Condition	none
Action	start CREATE-BR-RELATED-SELECTION-PANEL-CALLBACK (the bioreservoir superior to the workspace of the item, this window)

```
CREATE-BR-RELATED-SELECTION-PANEL-CALLBACK (BR: class bioreservoir, win: class window)
  if not (there exists a bioreservoir DBR such that (DBR is a-downstream-bioreservoir-of BR or DBR is an-
upstream-
    bioreservoir-of BR)) then go to exit-routine
  else
    create a message-dialog md, set the text of the message upon the SW of md to "Processing . . . . .  "
        and show the SW of md on the current window;
    panel = call uil-clone-grobj (MASTER-BR-MOLECULAR-QUERY-PANEL);
    SW = the subworkspace of panel;
    create a borderless-free-text FT ;
    change the text of FT to "[the name of BR]";
    transfer FT to SW at (60, 115);
    make panel permanent;
    call SET-RADIO-BOX-ID-PROC ("br-panel", "BR-RELATED-QUERY-PANEL-radio-box", SW, win);
    if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
      change the text of the callback of de to "DELETE-QUERY-PANEL-CALLBACK";
    show the SW of panel on win at (0, 0);
    change the text of the related-item of panel to "[the name of BR]";
  delete md;
 return;
exit-routine:
  create a message-dialog md, set the text of the message upon the SW of md to "There is no downstream
      bioreservoir of [the name of BR]", the symbol extra-large);
  return;
```

TABLE 139

```
SELECT-BR-RELATED-QUERY-PANEL-CALLBACK (button: class button, win : class window,
dlg: class query-panel)
BRN: symbol = the related-item of dlg;
SW: class kb-workspace = the subworkspace of dlg;
 create a message-dialog md, set the text of the message upon the SW of md to "Processing . . . . .  " and
    show the SW of md on the current window at (0, 0);
 rb-id = call GET-RADIO-BOX-ID-PROC ("br-panel", SW, win);
 type = call get-radio-box-value (dlg, rb-id);
 case (type) of
  function: MP = MASTER-BR-FUNCTION-QUERY-PANEL;
  function-subunits: MP = MASTER-BR-FUNCTION-SUBUNIT-QUERY-PANEL;
  function-domains: MP = MASTER-BR-FUNCTION-DOMAIN-QUERY-PANEL;
  function-motifs: MP = MASTER-BR-FUNCTION-MOTIF-QUERY-PANEL;
  function-groups: MP = MASTER-BR-FUNCTION-GROUP-QUERY-PANEL;
  function-domains-motifs: MP = MASTER-BR-FUNCTION-DOMAIN-MOTIF-QUERY-PANEL;
  function-location: MP = MASTER-BR-FUNCTION-LOCATION-QUERY-PANEL;
  function-location-subunits: MP = MASTER-BR-FUNCTION-LOCATION-SUBUNIT-QUERY-PANEL;
  function-location-domains: MP = MASTER-BR-FUNCTION-LOCATION-DOMAIN-QUERY-PANEL;
  function-location-motifs: MP = MASTER-BR-FUNCTION-LOCATION-MOTIF-QUERY-PANEL;
  function-location-groups: MP = MASTER-BR-FUNCTION-LOCATION-GROUP-QUERY-PANEL;
  function-location-domains-motifs: MP = MASTER-BR-FUNCTION-LOCATION-DOMAIN-MOTIF-
      QUERY-PANEL;
 panel = call clone-grobj (MP);
 change the text of the related-item of panel to "[BRN]";
 call SET-BR-RELATED-QUERY-PANEL-PROC (type, panel, win);
 show the subworkspace of dlg with its bottom right corner at the bottom right corner of the screen;
 show the subworkspace of panel at the center of the screen;
 delete md
```

TABLE 140

```
SET-BR-RELATED-QUERY-PANEL-PROC (type: symbol, panel: class query-panel, win : class window)
BRN: symbol = the related-item of panel:
SW: class kb-workspace = the subworkspace of panel:
 create a borderless-free-text FT and change the text of FT to "[BRN]" and transfer FT to SW at (175, 110):
 make panel permanent:
 if there exists a radio-button B upon SW such that (the id of B = "location") then
  call SET-RADIO-BOX-ID-PROC ("location", "0-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "direction") then
  call SET-RADIO-BOX-ID-PROC ("direction", "1-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "function") then
  call C-SET-RADIO-BOX-ID-PROC ("function", "2-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  call SET-RADIO-BOX-ID-PROC ("subunits", "3-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  call SET-RADIO-BOX-ID-PROC ("domains", "4-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  call SET-RADIO-BOX-ID-PROC ("motifs", "5-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  call SET-RADIO-BOX-ID-PROC ("groups", "6-BR-[type]-QUERY-radio-box", SW, win):
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
  change the text of the callback of de to "DELETE-QUERY-PANEL-CALLBACK":
 show panel on win at (0, 0):
 return:
```

TABLE 141

```
DOWNSTREAM-QUERY-LISTS-CALLBACK (button: class button, win : class window, panel: class query-panel)
BR: class bioreservoir = the bioreservoir named by the related-item of panel:
 create a message-dialog md, set the text of the message upon the SW of md to "Processing . . . . . " and
    show the SW of md on the current window:
  call DOWNSTREAM-FUNCTION-QUERY-LISTS-PROC (BR, panel, win):
 delete md:
 return
```

TABLE 142

```
DOWNSTREAM-FUNCTION-QUERY-LISTS-PROC (BR: class bioreservoir, panel: class c-query-panel, win : class window)
  BRN: symbol = the name of BR:
  DAPL = call CREATE-BR-QUERY-LIST-PROC ("DAPL", panel, win):
  DAPBRL = call CREATE-BR-QUERY-LIST-PROC ("DAPBRL", panel,win):
  DEFL = call CREATE-BR-QUERY-LIST-PROC ("DEFL", panel, win):
  DEFBRL = call CREATE-BR-QUERY-LIST-PROC ("DEFBRL", panel,win):
  DCRL = call CREATE-BR-QUERY-LIST-PROC ("DCRL", panel, win):
  DCRBRL = call CREATE-BR-QUERY-LIST-PROC ("DCRBRL", panel,win).
  DPKL = call CREATE-BR-QUERY-LIST-PROC ("DPKL", panel, win):
  DPKBRL = call C-CREATE-BR-QUERY-LIST-PROC ("DPKBRL", panel, win).
  DPPL = call CREATE-BR-QUERY-LIST-PROC ("DPPL", panel, win):
  DPPBRL = call CREATE-BR-QUERY-LIST-PROC ("DPPBRL", panel,win).
  DTFL = call CREATE-BR-QUERY-LIST-PROC ("DTFL", panel, win):
  DTFBRL = call CREATE-BR-QUERY-LIST-PROC ("DTFBRL", panel, win):
  DGL = call CREATE-BR-QUERY-LIST-PROC ("DGL", panel, win):
  DGBRL = call C-CREATE-BR-QUERY-LIST-PROC ("DGBRL", panel, win).
  for DBR = each bioreservoir that is a-downstream-bioreservoir-of BR do
   if the biopool upon the subworkspace of DBR exists then insert "[the name of DBR]" at the end of the text list DAPBRL:
     PO = the biopool upon the subworkspace of DBR:
     if there exists a bioreactant connected to PO then
      for R = each bioreactant connected to PO do
       if R is a extracell-ligand.r or R is a extracell-antagonist.r or R is a ligand.r or R is a antagonist.r then
          insert "[the name of DBR]" at the end of the text list DEFBRL:
       if R is a cell-receptor.r or R is a receptor.r then insert "[the name of DBR]" at the end of the text list DCRBRL:
       if R is a prot.kinase.r then insert "[the name of DBR]" at the end of the text list DPKBRL.
       if R is a prot-phosphatase.r then insert "[the name of DBR]" at the end of the text list DPPBRL:
       if R is a transcription-factor.r then insert "[the name of DBR]" at the end of the text list DTFBRL:
       if R is a gene.r then insert "[the name of DBR]" at the end of the text list DGBRL:
       {if R is a mRNA.r then insert "[the name of DBR]" at the end of the text list DMRBRL:}
  call FILL-BRLET-LIST-PROC (DAPBRL, DAPL, panel, win):
  call FILL-BRLET-LIST-PROC (DEFBRL, DEFL, panel, win):
  call FILL-BRLET-LIST-PROC (DCRBRL, DCRL, panel, win):
  call FILL-BRLET-LIST-PROC (DPKBRL, DPKL, panel, win):
  call FILL-BRLET-LIST-PROC (DPPBRL, DPPL, panel, win):
  call FILL-BRLET-LIST-PROC (DTFBRL, DTFL, panel, win):
  call FILL-BRLET-LIST-PROC (DGBRL, DGL, panel, win):
  return
```

TABLE 143

```
CREATE-BR-QUERY-LIST-PROC (lbl: text, panel: class query-panel, win: class window) = (class scroll-text-list)
PN: symbol = the name of panel;
BRN: symbol = symbol (the related-item of panel);
 if BRN is not none then
   if not (there exists a scroll-text-list L such that (the label of L = "[PN]-[lbl]-[BRN]")) then
     create a scroll-text-list L;
     conclude that the label of L = "[PN]-[lbl]-[BRN]";
     transfer L to the subworkspace of safe-home-for-lists at (-105, 25);
     conclude that L is a-list-associated-to-dlg panel;}
     return L;
   else
    return L
```

TABLE 144

```
FILL-BR.ET-LIST-PROC (BRL: class scroll-text-list, L: class scroll-text-list, panel: class query-panel, win : class
window)
 move BRL to (135, 10);
 if the number of elements in BRL >= 0 then
  for T = each text in BRL do
   BRN = symbol (T);
   BR = the bioreservoir named by BRN;
   if the ref-bioentity of BR has a current value then
     EN = the bioentity named by the ref-bioentity of BR;
     insert "[the name of EN]" at the end of the text list L;
 return
```

TABLE 145

```
QUERY-BR-RELATED-CALLBACK (button: class button, win : class window, panel: class query-panel)
BRN: symbol = symbol (the related-item of panel);
PN: symbol = the name of panel;
SW: class kb-workspace = the subworkspace of panel;
  if not (there exists a scroll-text-list L such that (the label of L = "[PN]-DPKL-[BRN]" or
       the label of L = "[PN]-UPKL-[BRN]" )) then
    create a message-dialog md, set the text of the message upon the SW of md to "Required information is not available.
      Click on Downstream or Upstream and try to query again" and show the SW of md on the current window at (0, 0);
  else
    create a message-dialog md, set the text of the message upon the SW of md to "Processing . . . . . " and
        show the SW of md on the current window at (0, 0);
 rb-id0 = call GET-RADIO-BOX-ID-PROC ("direction", SW, win);
 type0 = call get-radio-box-value (panel, rb-id0);
 if there exists a radio-button B upon SW such that (the id of B = "function") then
   rb-id2 = call GET-RADIO-BOX-ID-PROC ("function", SW, win);
   type2 = call get-radio-box-value (panel, rb-id2);
   FL, FBRL = call GET-BR-FUNCTION-LIST-PROC (type0, type2, panel, win);
 if FBRL exists then
   create a panel-text FT1;
   TX1 = capitalize-words ("bioReservoirs [type0] of [BRN] that contain [type2]");
 change the text of FT1 to TX1;
 create a panel-text FT2;
 if FL exists then
   if not (there exists a radio-button B upon SW such that (the id of B = "subunits" or the id of B = "domains" or
       the id of B = "motifs" or the id of B = "groups")) then
     if there exists a scroll-text-list EL such that (the label of EL = "[PN]-[type0]-[type2]-ETL") then
       ETL = EL
     else
       create a c-scroll-text-list ETL;
       conclude that the label of ETL = "[PN]-[type0]-[type2]-ETL";
       conclude that ETL is a-list-associated-to-dlg panel;
       transfer ETL to the subworkspace of safe-home-for-lists at (50,-245);
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]");
       for T1 = each text in FL do
         insert T1 at the end of the text list ETL;
   else
    ETL, TX2 = call QUERY-BR-STRUCTURE-LISTS-PROC (type0, type2, FL, panel, win);
 change the text of FT2 to "[TX2]";
 call CREATE-DOUBLE-QUERY-OUTPUT-PANEL-PROC (FBRL, ETL, FT1, FT2, panel, win);
 delete md;
 return;
```

TABLE 146

```
GET-BR-FUNCTION-LIST-PROC (type2: symbol. type3: symbol. panel:
class query-panel. win : class window) = (class scroll-text-list. class scroll-text-list)

PN: symbol = the name of panel:
BRN: symbol = the related-item of panel:
 if type2 is downstream then
    case (type3) of
     extracellular-factors:
        lbl = "[PN]-DEFL-[BRN]":
        lblbr = "[PN]-DEFBRL-[BRN]":
     cell-receptors:
        lbl = "[PN]-DCRL-[BRN]":
        lblbr = "[PN]-DCRBRL-[BRN]":
     protein-kinases:
        lbl = "[PN]-DPKL-[BRN]":
        lblbr = "[PN]-DPKBRL-[BRN]":
     protein-phosphatases:
        lbl = "[PN]-DPPL-[BRN]":
        lblbr = "[PN]-DPPBRL-[BRN]":
     transcription-factors:
        lbl = "[PN]-DTFL-[BRN]":
        lblbr = "[PN]-DTFBRL-[BRN]":
     genes:
        lbl = "[PN]-DGL-[BRN]":
        lblbr = "[PN]-DGBRL-[BRN]":
     any-protein:
        lbl = "[PN]-DAPL-[BRN]":
        lblbr = "[PN]-DAPBRL-[BRN]":
     no-function-selection:
        lbl = "[PN]-DNL-[BRN]":
        lblbr = "[PN]-DNBRL-[BRN]": else
  if type2 is upstream then
    case (type3) of
     extracellular-factors:
        lbl = "[PN]-UEFL-[BRN]":
        lblbr = "[PN]-UEFBRL-[BRN]":
     cell-receptors:
        lbl = "[PN]-UCRL-[BRN]":
        lblbr = "[PN]-UCRBRL-[BRN]":
     protein-kinases:
        lbl = "[PN]-UPKL-[BRN]":
        lblbr = "[PN]-UPKBRL-[BRN]":
     protein-phosphatases:
        lbl = "[PN]-UPPL-[BRN]":
        lblbr = "[PN]-UPPBRL-[BRN]":
     transcription-factors:
        lbl = "[PN]-UTFL-[BRN]":
        lblbr = "[PN]-UTFBRL-[BRN]":
     genes:
        lbl = "[PN]-UGL-[BRN]":
        lblbr = "[PN]-UGBRL-[BRN]":
     any-protein:
        lbl = "[PN]-UAPL-[BRN]":
        lblbr = "[PN]-UAPBRL-[BRN]":
     no-function-selection:
        lbl = "[PN]-UNL-[BRN]":
        lblbr = "[PN]-UNBRL-[BRN]":
  if there exists a c-scroll-text-list FL such that (the label of FL = lbl) then
     move FL to (-110. -280):
  if there exists a c-scroll-text-list FBRL such that (the label of FBRL = lblbr) then
     move FL to (-30. -280) :
  return FL. FBRL.
```

TABLE 147

```
QUERY-BR-STRUCTURE-LISTS-PROC (type0: symbol, type2: symbol, FL: class scroll-text-list,
panel: class query-panel, win : class window) = (class scroll-text-list, text)
  PN: symbol = the name of panel;
  SW: class kb-workspace = the subworkspace of panel;
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  type3, PSA = call FIND-SPECIFIC-ARRAY-PROC ("subunits", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  type4, PSA-2 = call FIND-SPECIFIC-ARRAY-PROC ("subunits-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  type5, PDA = call FIND-SPECIFIC-ARRAY-PROC ("domains", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  type6, PDA-2 = call FIND-SPECIFIC-ARRAY-PROC ("domains-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  type7, PMA = call FIND-SPECIFIC-ARRAY-PROC ("motifs", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  type8, PMA-2 = call FIND-SPECIFIC-ARRAY-PROC ("motifs-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  type9, PGA = call FIND-SPECIFIC-ARRAY-PROC ("groups", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  type10, PGA-2 = call FIND-SPECIFIC-ARRAY-PROC ("groups-2", panel, win);
 if PSA exists then
    if PSA-2 exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type3]-[type4]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol subunit,
            the symbol subunit, type0, type2, type3, type4, FL, PSA, PSA-2, panel, win);
    else
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type3]-ETL") then
         go to final-routine
       else
      ETL, TX2  = call MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol subunit, type0, type2, type3,
          FL, PSA, panel, win);
 else
 if PDA exists then
    if PDA-2 exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type5]-[type6]-ETL")
then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
            the symbol domain, type0, type2, type5, type6, FL, PDA, PDA-2, panel, win)
    else
    if PMA exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type5]-[type7]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
            the symbol motif, type0, type2, type5, type7, FL, PDA, PMA, panel, win)
    else
    if PGA exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type5]-[type9]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
            the symbol group, type0, type2, type5, type9, FL, PDA, PGA, panel, win)
    else
      if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type5]-ETL") then
         go to final-routine
       else
      ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-PROC  (the symbol  domain, type0, type2, type5,
          FL, PDA, panel, win);
 else
 if PMA exists then
    if PMA-2 exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type7]-[type8]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol motif,
            the symbol motif, type0, type2, type7, type8, FL, PMA, PMA-2, panel, win)
```


continue

TABLE 147 cont.

```
  else
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type7]-ETL") then
      go to final-routine
    else
  if PGA exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type7]-[type9]-ETL") then
      go to final-routine
    else
      ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol motif,
        the symbol group, type0, type2, type7, type9, FL, PMA, PGA, panel, win)
  else
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-ETL") then
      go to final-routine
    else
    ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol motif, type0, type2, type7,
      FL, PMA, panel, win);
else
if PGA exists then
  if PGA-2 exists then
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type9]-[type10]-ETL") then
      go to final-routine
    else
      ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol group,
        the symbol group, type0, type2, type9, type10, FL, PGA, PGA-2, panel, win)
  else
    if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type9]-ETL") then
      go to final-routine
    else
    ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol group, type0, type2, type9,
      FL, PGA, panel, win);
return ETL, TX2;
final-routine:
  return ETL, TX2;
```

TABLE 148

```
MERGE-BR-FUNCTION-STRUCTURE-PROC (tag1: symbol, type0: symbol, type2: symbol, type3: symbol,
FL: class scroll-text-list, QA: class query-array, panel: class query-panel, win : class window) =
(class scroll-text-list text)

PN: symbol = the name of panel;
BRN: symbol = the related-item of panel;
 if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type3]-ETL") then
  go to exit-routine
 else
    create a scroll-text-list ETL;
    conclude that the label of ETL = "[PN]-[type0]-[type2]-[type3]-ETL";
    conclude that ETL is a-list-associated-to-dlg panel;
    transfer ETL to the subworkspace of safe-home-for-lists at (50, -245);
    TX2 = the scroll-title of ETL;
    if the number of elements in FL = 0 then
      TX2 = capitalize-words ("There are no structures defined in this module of [type2] that are [type0] of [BRN]");
 else
   call BR-FILL-ETL-LIST-PROC (FL, QA, ETL, panel, win);
   if the number of elements in ETL >= 0 then
     TX2 = capitalize-words ("[type2] that are [type0] of [BRN] and contain [type3]");
   else
     if the number of elements in ETL = 0 then
       if type3 = symbol ("no-[tag1]-selection") then
         TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]")
       else
         TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]. None
             of those contain [type3]");
       for T1 = each text in FL do
        insert T1 at the end of the text list ETL;
  return ETL, TX2;
  exit-routine:
    TX2 = the scroll-title of ETL;
    return ETL, TX2;
```

TABLE 149

```
MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (tag1: symbol, tag2: symbol, type0: symbol, type2:
symbol, type4: symbol, type5: symbol, FL: class scroll-text-list, DQA: class query-array, MQA: class query-array,
panel: class query-panel, win : class window) = (class scroll-text-list, text)
PN: symbol = the name of panel:
BRN: symbol = symbol (the related-item of panel):
 if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type2]-[type4]-[type5]-ETL") then
    go to exit-routine
 else
    create a scroll-text-list ETL:
    conclude that the label of ETL = "[PN]-[type0]-[type2]-[type4]-[type5]-ETL":
    conclude that ETL is a-list-associated-to-dlg panel:
    transfer ETL to the subworkspace of safe-home-for-lists at (50, -245):
    TX2 = the scroll-title of ETL:
    if the number of elements in FL = 0 then
      TX2 = capitalize-words ("There are no structures defined in this module of [type2] that are [type0] of [BRN]") :
    else
      if the array-length of DQA = 0 then
        if the array-length of MQA = 0 then
          if type5 = symbol ("no-[tag2]-selection") then
            TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN].
                None of those contain [type4]")
          else
            if type4 = symbol ("no-[tag1]-selection") then
              TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN].
                  None of those contain [type5]")
            else
              if type5 = symbol ("no-[tag2]-selection") and type4 = symbol ("no-[tag1]-selection") then
                TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]")
              else
                TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN].
                  None of those contain [type4] or [type5]"):
          for T1 = each text in FL do
          insert T1 at the end of the text list ETL:
      else
        if type4 = symbol ("no-[tag1]-selection") then
          TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN] and
            contain [type5]")
        else
          TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN] and
            contain [type5]. None of those contain [type4]"):
        call BR-FILL-ETL-LIST-PROC (FL, MQA, ETL, panel, win):
 else
   if the array-length of MQA = 0 then
     if type5 = symbol ("no-[tag2]-selection") then
       TX2 = capitalize-words ("[type2] with structures defined n this module that are [type0] of [BRN] and
           contain [type4]")
     else
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN] and
           contain [type4]. None of those contain [type5]"):
       call BR-FILL-ETL-LIST-PROC (FL, DQA, ETL, panel, win):
   else
     TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN] and
           contain [type4] and [type5]"):
   call BR-FILL-DOUBLE-ETL-LIST-PROC (FL, DQA, MQA, ETL, panel, win):
 return ETL, TX2:
exit-routine:
   TX2 = the scroll-title of ETL:
   return ETL, TX2:
```

TABLE 150

```
BR-FILL-ETL-LIST-PROC (FL: class scroll-text-list, QA: class query-array, ETL: class scroll-text-list, panel: class
query-panel, win : class window)
 if the number of elements in FL >= 0 and the array-length of QA >= 0 then
   for T = each text in FL do
     for AE = each symbol in QA do
       if T = "[AE]" then insert T at the end of the text list ETL:
 return
```

TABLE 153

```
DOWNSTREAM-FUNCTION-LOCATION-CALLBACK (button: class button.win: class window.
panel: class query-panel)
  BR = the bioreservoir named by the related-item of panel:
  create a message-dialog md.  set the text of the message upon the SW of md to "Processing . . . . .  "
    and show the SW of md on the current window:
  call DOWNSTREAM-FUNCTION-QUERY-LISTS-PROC (BR. panel. win):
  call DOWNSTREAM-LOCATION-QUERY-LISTS-PROC (BR. panel. win):
  delete md:
  return:
```

TABLE 154

```
DOWNSTREAM-LOCATION-QUERY-LISTS-PROC (BR: class bioreservoir. panel: class query-panel.
win : class window)
  BRN = the name of BR:
  DACL = call CREATE-BR-QUERY-LIST-PROC ("DACL". panel. win):
  DACBRL = call CREATE-BR-QUERY-LIST-PROC ("DACBRL". panel. win):
  DECL = call CREATE-BR-QUERY-LIST-PROC ("DECL". panel. win):
  DECBRL = call CREATE-BR-QUERY-LIST-PROC ("DECBRL". panel.win):
  DCML = call CREATE-BR-QUERY-LIST-PROC ("DCML". panel. win):
  DCMBRL = call CREATE-BR-QUERY-LIST-PROC ("DCMBRL". panel.win):
  DCYL = call CREATE-BR-QUERY-LIST-PROC ("DCYL". panel. win):
  DCYBRL = call CREATE-BR-QUERY-LIST-PROC ("DCYBRL". panel.win):
  DNUL = call CREATE-BR-QUERY-LIST-PROC ("DNUL". panel. win):
  DNUBRL = call CREATE-BR-QUERY-LIST-PROC ("DNUBRL". panel. win):
  DERL = call CREATE-BR-QUERY-LIST-PROC ("DERL". panel. win):
  DERBRL = call CREATE-BR-QUERY-LIST-PROC ("DERBRL". panel.win):
  DGOL = call CREATE-BR-QUERY-LIST-PROC ("DGOL". panel. win):
  DGOBRL = call CREATE-BR-QUERY-LIST-PROC ("DGOBRL". panel.win):
  DENL = call CREATE-BR-QUERY-LIST-PROC ("DENL". panel. win):
  DENBRL = call CREATE-BR-QUERY-LIST-PROC ("DENBRL". panel.win):
  for DBR = each bioreservoir that is a-downstream-bioreservoir-of BR do
    insert "[the name of DBR]" at the end of the text list DACBRL:
    if DBR is a ext-sol-mol-reservoir then insert "[the name of DBR]" at the end of the text list DECBRL else
      CL = call C-FIND-COMPARTMENT-PROC (DBR. panel. win):
      if CL = the symbol cell-membrane then insert "[the name of DBR]" at the end of the text list DCMBRL else
      if CL = the symbol cytosol then insert "[the name of DBR]" at the end of the text list DCYBRL else
      if CL = the symbol nucleus then insert "[the name of DBR]" at the end of the text list DNUBRL else
      if CL = the symbol e-reticulum then insert "[the name of DBR]" at the end of the text list DERBRL else
      if CL = the symbol golgi then insert "[the name of DBR]" at the end of the text list DGOBRL else
      if CL = the symbol endosomes then insert "[the name of DBR]" at the end of the text list DENBRL:
  call FILL-BR.ET-LIST-PROC (DACBRL. DACL. panel. win):
  call FILL-BR.ET-LIST-PROC (DECBRL. DECL. panel. win):
  call FILL-BR.ET-LIST-PROC (DCMBRL. DCML. panel. win):
  call FILL-BR.ET-LIST-PROC (DCYBRL. DCYL. panel. win):
  call FILL-BR.ET-LIST-PROC (DNUBRL. DNUL. panel. win):
  call FILL-BR.ET-LIST-PROC (DERBRL. DERL. panel. win):
  call FILL-BR.ET-LIST-PROC (DGOBRL. DGOL. panel. win):
  call FILL-BR.ET-LIST-PROC (DENBRL. DENL. panel. win).
  return:
```

TABLE 155

```
FIND-COMPARTMENT-PROC (BR: class bioreservoir. panel: class query-panel. win : class window) = (symbol )
  RH = the bioreservoir-bin superior to the workspace of BR:
  BM1 = the biomodel superior to the workspace of RH:
  if BM1 is a cell-compartment then CO = BM1 else
    BM2 = the biomodel superior to the workspace of BM1:
    if BM2 is a cell-compartment then CO = BM2 else
      BM3 = the biomodel superior to the workspace of BM2:
      if BM3 is a cell-compartment then CO = BM3 else
        BM4 = the biomodel superior to the workspace of BM3:
        if BM4 is a cell-compartment then CO = BM4 else
          BM5 = the biomodel superior to the workspace of BM4:
          if BM5 is a cell-compartment then CO = BM5:
  CL = the class of CO:
  return CL:
```

TABLE 156

```
QUERY-BR-RELATED-LOCATION-CALLBACK (button: class button, win : class window, panel: class query-panel)
BRN: symbol = symbol (the related-item of panel);
PN: symbol = the name of panel;
SW: class workspace = the subworkspace of panel;
 if not (there exists a scroll-text-list L such that (the label of L = "[PN]-DPKL-[BRN]" or the label of L =
    "[PN]-UPKL-[BRN]" )) then
  create a message-dialog md.set the text of the message upon the SW of md to "Required information is not available.
    Click on Downstream or Upstream and try to query again" and show the SW of md on the current window
 else
    create a message-dialog md. set the text of the message upon the SW of md to "Processing . . . . .  " and
    show the SW of md on the current window;
 create a panel-text FT1;
 create a panel-text FT2;
 if not (there exists a radio-button B upon SW such that (the id of B = "subunits" or the id of B = "domains" or
     the id of B = "motifs" or the id of B = "groups")) then
  BRL, ETL, TX1, TX2 = call MERGE-LOCATION-FUNCTION-ONLY-PROC (panel, win)
 else
  type0, type1, type2, LL, FL, BRL,TX1 = call MERGE-LOCATION-FUNCTION-PROC (panel, win);
  if there exists a radio-button B1 upon SW such that (the id of B1 = "location") and there exists a uil-radio-button B2
      upon SW such that the id of B2 = "function") then
    PETL = call MERGE-LL-FL-PROC (type1,type2, LL, FL, panel,win);
    ETL, TX2 = call QUERY-BR-LOCATION-FUNCTION-STRUCTURE-LISTS-PROC (type0, type1, type2,
      PETL, panel, win)
   else
   if there exists a radio-button B upon SW such that (the id of B = "location") then
    ETL, TX2 = call QUERY-BR-LOCATION-STRUCTURE-LISTS-PROC (type0, type1, LL, panel, win)
   else
   if there exists a radio-button B upon SW such that (the id of B = "function") then
    ETL, TX2 = call QUERY-BR-FUNCTION-STRUCTURE-LISTS-PROC (type0, type2, FL, panel, win);
 change the text of FT1 to "[TX1]";
 change the text of FT2 to "[TX2]";
 call CREATE-DOUBLE-QUERY-OUTPUT-PANEL-PROC (BRL, ETL, FT1, FT2, panel, win);
 delete md;
 return;
```

TABLE 157

```
MERGE-LOCATION-FUNCTION-ONLY-PROC  (panel: class query-panel, win : class window) =
(class scroll-text-list, class scroll-text-list, text, text)
BRN: symbol = symbol (the related-item of panel);
PN: symbol = the name of panel;
SW: class kb-workspace = the subworkspace of panel;
 if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-ETL-[BRN]") then
  for ETL = each scroll-text-list do
   if the label of ETL = "[PN]-ETL-[BRN]" then delete ETL;
 create a scroll-text-list ETL;
 conclude that the label of ETL = "[PN]-ETL-[BRN]";
 conclude that ETL is a-list-associated-to-dlg panel;
 transfer ETL to the subworkspace of safe-home-for-lists at (50, -245);
 rb-id0 = call GET-RADIO-BOX-ID-PROC ("direction", SW, win);
 type0 = call uil-get-radio-box-value (panel, rb-id0);
 rb-id1 = call GET-RADIO-BOX-ID-PROC ("location", SW, win);
 type1 = call uil-get-radio-box-value (panel, rb-id1);
 LL, LBRL = call GET-BR-LOCATION-LIST-PROC (type0, type1, panel, win);
 rb-id2 = call GET-RADIO-BOX-ID-PROC ("function", SW, win);
 type2 = call uil-get-radio-box-value (panel, rb-id2);
 FL, FBRL = call GET-BR-FUNCTION-LIST-PROC (type0, type2,  panel, win);
 if there exists a scroll-text-list BRL such that (the label of BRL = "[PN]-BRL-[BRN]") then
  for BRL = each c-scroll-text-list do
   if the label of BRL = "[PN]-BRL-[BRN]" then delete BRL;
 create a scroll-text-list BRL;
 conclude that the label of BRL = "[PN]-BRL-[BRN]";
 conclude that BRL is a-list-associated-to-dlg panel;
 transfer BRL to the subworkspace of safe-home-for-lists at (135, -235);
 TX1 = call MERGE-BRL-FUNCTION-LOCATION-PROC (type0, type1, type2, LBRL, FBRL, BRL, panel, win);
 TX2 = call MERGE-ETL-FUNCTION-LOCATION-ONLY-PROC (type0, type1, type2, LL, FL, ETL, panel, win);
 return BRL, ETL, TX1, TX2;
end
```

TABLE 158

```
MERGE-LL-FL-PROC (LL: class c-scroll-text-list. FL: class scroll-text-list.
  panel: class query-panel. win : class window) = (class scroll-text-list)
PN: symbol = the name of panel:
BRN: symbol = symbol (the related-item of panel):
 if there exists a c-scroll-text-list PETL such that (the label of PETL = "[PN]-PETL-[BRN]") then
  for PETL = each c-scroll-text-list do
   if the label of PETL = "[PN]-PETL-[BRN]" then delete PETL:
 create a c-scroll-text-list PETL:
 conclude that the label of PETL = "[PN]-PETL-[BRN]":
 {conclude that PETL is a-list-associated-to-dlg panel;}
 transfer PETL to the subworkspace of safe-home-for-lists at (50. -245):
 for TF = each text in FL do
  for TL = each text in LL do
   if TF = TL then insert TF at the end of the text list PETL:
 return PETL:
```

TABLE 159

```
GET-BR-LOCATION-LIST-PROC (type0: symbol. type1: symbol. panel: class
  query-panel. win : class window) = (class scroll-text-list. class scroll-text-list)
PN = the name of panel:
BRN = the related-item of panel:
 if type0 is downstream then
   case (type1) of
    extracellular:
       lbl = "[PN]-DECL-[BRN]";   lblbr = "[PN]-DECBRL-[BRN]";
    cell-membrane:
       lbl = "[PN]-DCML-[BRN]";   lblbr = "[PN]-DCMBRL-[BRN]";
    cytosol:
       lbl = "[PN]-DCYL-[BRN]";   lblbr = "[PN]-DCYBRL-[BRN]";
    nucleus:
       lbl = "[PN]-DNUL-[BRN]";   lblbr = "[PN]-DNUBRL-[BRN]";
    e-reticulum:
       lbl = "[PN]-DERL-[BRN]";   lblbr = "[PN]-DERBRL-[BRN]";
    golgi:
       lbl = "[PN]-DGOL-[BRN]";   lblbr = "[PN]-DGOBRL-[BRN]";
    endosomes:
       lbl = "[PN]-DENL-[BRN]";   lblbr = "[PN]-DENBRL-[BRN]";
    no-location-selection :
       lbl = "[PN]-DNL-[BRN]";    lblbr = "[PN]-DNBRL-[BRN]";
   else
  if type0 is upstream then
    case (type1) of
    extracellular:
       lbl = "[PN]-UECL-[BRN]";   lblbr = "[PN]-UECBRL-[BRN]";
    cell-membrane:
       lbl = "[PN]-UCML-[BRN]";   lblbr = "[PN]-UCMBRL-[BRN]";
    cytosol:
       lbl = "[PN]-UCYL-[BRN]";   lblbr = "[PN]-UCYBRL-[BRN]";
    nucleus:
       lbl = "[PN]-UNUL-[BRN]";   lblbr = "[PN]-UNUBRL-[BRN]";
    e-reticulum:
       lbl = "[PN]-UERL-[BRN]";   lblbr = "[PN]-UERBRL-[BRN]";
    golgi:
       lbl = "[PN]-UGOL-[BRN]";   lblbr = "[PN]-UGOBRL-[BRN]";
    endosomes:
       lbl = "[PN]-UENL-[BRN]";   lblbr = "[PN]-UENBRL-[BRN]";
    no-location-selection :
       lbl = "[PN]-UNL-[BRN]";    lblbr = "[PN]-UNBRL-[BRN]";
 if there exists a scroll-text-list LL such that (the label of LL = lbl) then move LL to (-110. -280):
 if there exists a scroll-text-list LBRL such that (the label of LBRL = lblbr) then move LBRL to (-30. 30):
 return LL. LBRL:
```

TABLE 160

```
MERGE-BRL-FUNCTION-LOCATION-PROC (type0: symbol. type1: symbol.
type2: symbol. LBRL: class scroll-text-list. FBRL: class scroll-text-list. ETL: class
scroll-text-list. panel: class query-panel. win : class window) = (text)

BRN: symbol = symbol (the related-item of panel):
 if the number of elements in FBRL = 0 and the number of elements in LBRL = 0
   then
      if type2 = the symbol no-function-selection then
       TX2 = capitalize-words ("there are no bioReservoirs
[type0] of
[BRN]
and located in the [type1]")
      else
      if type1 = the symbol no-location-selection then
       TX2 = capitalize-words ("There are no bioReservoirs
of [type2] [type0] of
[BRN]
and located in the [type1]")
     else
 if the number of elements in FBRL = 0 and the number of elements in LBRL >= 0
   then
      if type2 = the symbol no-function-selection then
       TX2 = capitalize-words ("bioReservoirs [type0] of
[BRN]
and located in the [type1]")
       else
        TX2 = capitalize-words ("bioReservoirs [type0] of
[BRN]
and located in the [type1].
 None are [type2]"):
      for TL = each text in LBRL do
       insert TL at the end of the text list ETL:
     else
 if the number of elements in FBRL >= 0 and the number of elements in LBRL = 0
   then
    if type1 = the symbol no-location-selection then
     TX2 = capitalize-words ("bioReservoirs of [type2] [type0] of
[BRN]")
     else
      TX2 = capitalize-words ("bioReservoirs of [type2] [type0] of
[BRN].
 None of those are located in the [type1]"):
    for TF = each text in FBRL do
     insert TF at the end of the text list ETL:
  else
 if the number of elements in FBRL >= 0 and the number of elements in LBRL >= 0
   then
    TX2 = capitalize-words ("bioReservoirs of [type2] [type0] of
[BRN]
and located in the [type1]"):
    for TF = each text in FBRL do
     for TL = each text in LBRL do
      if TF = TL then
       insert TF at the end of the text list ETL:
 return TX2:
```

TABLE 161

```
MERGE-ETL-FUNCTION-LOCATION-ONLY-PROC (type0: symbol, type1: symbol, type2: symbol, LL: class c-
scroll-text-list, FL: class c-scroll-text-list, ETL: class scroll-text-list, panel: class query-panel, win : class window) =
(text)

BRN: symbol = symbol (the related-item of panel):
 if the number of elements in FL = 0 and the number of elements in LL = 0
   then
    if type2 = the symbol no-function-selection then
     TX2 = capitalize-words ("there are no entities with structures defined in this module that
        are [type0] of [BRN]and located in the [type1]")
    else
    if type1 = the symbol no-location-selection then
     TX2 = capitalize-words ("There are no [type2] with structures defined in this module of
        that are [type0] of [BRN]")
     end else
 if the number of elements in FL = 0 and the number of elements in LL >=0
   then
      if type2 = the symbol no-function-selection then
       TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN]
          and located in the [type1]")
      else
       TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN]
          and located in the [type1]. None are [type2]"):
    for TL = each text in LL do
     insert TL at the end of the text list ETL:
   else
  if the number of elements in FL >= 0 and the number of elements in LL =0 then
    if type1 = the symbol no-location-selection then
     TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] [BRN]")
    else
     TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN].
        None of those are located in the [type1]"):
    for TF = each text in FL do
     insert TF at the end of the text list ETL:
    else
 if the number of elements in FL>= 0 and the number of elements in LL>= 0 then
    TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
        and located in the [type1]"):
    for TF = each text in FL do
     for TL = each text in LL do
      if TF = TL then
       insert TF at the end of the text list ETL:
return TX2:
```

TABLE 162

```
MERGE-LOCATION-FUNCTION-PROC, (panel: class query-panel, win : class window) = (symbol, symbol, symbol,
class scroll-text-list, class scroll-text-list, class scroll-text-list, text)
BRN: symbol = symbol (the related-item of panel):
PN: symbol = the name of panel:
SW: class kb-workspace = the subworkspace of panel:
 rb-id0 = call GET-RADIO-BOX-ID-PROC ("direction", SW, win):
 type0 = call uil-get-radio-box-value (panel, rb-id0):
 rb-id1 = call GET-RADIO-BOX-ID-PROC ("location", SW, win):
 type1 = call uil-get-radio-box-value (panel, rb-id1):
 LL, LBRL = call GET-BR-LOCATION-LIST-PROC (type0, type1, panel, win):
 rb-id2 = call GET-RADIO-BOX-ID-PROC ("function", SW, win):
 type2 = call uil-get-radio-box-value (panel, rb-id2):
 FL, FBRL = call GET-BR-FUNCTION-LIST-PROC (type0, type2, panel, win):
 if there exists a scroll-text-list BRL such that (the label of BRL = "[PN]-BRL-[BRN]") then
  for BRL = each scroll-text-list do
   if the label of BRL = "[PN]-BRL-[BRN]" then delete BRL:
 create a scroll-text-list BRL:
 conclude that the label of BRL = "[PN]-BRL-[BRN]":
 conclude that BRL is a-list-associated-to-dlg panel:
 transfer BRL to the subworkspace of safe-home-for-lists at (135, -235):
 TX1 = call C-MERGE-BRL-FUNCTION-LOCATION-PROC (type0, type1, type2, LBRL, FBRL, BRL, panel, win):
return type0, type1, type2, LL, FL, BRL, TX1:
```

TABLE 163

```
QUERY-BR-LOCATION-STRUCTURE-LISTS-PROC (type0: symbol, type1: symbol,
  PETL: class scroll-text-list, panel: class query-panel, win : class window) = (class scroll-text-list, text)
PN: symbol = the name of panel;
SW: class kb-workspace = the subworkspace of panel;
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  type3, PSA = call FIND-SPECIFIC-ARRAY-PROC ("subunits", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  type4, PSA-2 = call FIND-SPECIFIC-ARRAY-PROC ("subunits-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  type5, PDA = call FIND-SPECIFIC-ARRAY-PROC ("domains", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  type6, PDA-2 = call FIND-SPECIFIC-ARRAY-PROC ("domains-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  type7, PMA = call FIND-SPECIFIC-ARRAY-PROC ("motifs", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  type8, PMA-2 = call FIND-SPECIFIC-ARRAY-PROC ("motifs-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = I-= "groups") then
  type9, PGA = call FIND-SPECIFIC-ARRAY-PROC ("groups", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  type10, PGA-2 = call FIND-SPECIFIC-ARRAY-PROC ("groups-2", panel, win);
 if PSA exists then
    if PSA-2 exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type3]-[type4]-ETL") then
        go to final-routine
       else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol subunit,
          the symbol subunit, type0, type1, type3, type4, PETL, PSA, PSA-2, panel, win);
     else
      ETL, TX2  = call MERGE-BR-LOCATION-STRUCTURE-PROC (the symbol subunit, type0, type1, type3,
        PETL, PSA, panel, win);
  else
  if PDA exists then
    if PDA-2 exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type5]-[type6]-ETL") then
        go to final-routine
       else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
          the symbol domain, type0, type1, type5, type6, PETL, PDA, PDA-2, panel, win)
     else
    if PMA exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type5]-[type7]-ETL") then
        go to final-routine
       else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
          the symbol motif, type0, type1, type5, type7, PETL, PDA, PMA, panel, win)
     else
    if PGA exists then
       if there exists a scroll-text-list ETL such that (the label of  ETL = "[PN]-[type0]-[type1]-[type5]-[type9]-ETL") then
        go to final-routine
       else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol domain,
          the symbol group, type0, type1, type5, type9, PETL, PDA, PGA, panel, win)
     end else
      ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-PROC  (the symbol domain, type0, type1, type5,
        PETL, PDA, panel, win);
  else
  if PMA exists then
    if PMA-2 exists then
       if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type7]-[type8]-ETL") then
        go to final-routine
       else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol motif,
          the symbol motif, type0, type1, type7, type8, PETL, PMA, PMA-2, panel, win
     else
```

continue

TABLE 163 cont.

```
    if PGA exists then
      if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-type1]-[type7]-[type9]-ETL") then
        go to final-routine
      else
        ETL, TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbo. motif.
          the symbol group. type0. type1. type7. type9. PETL. PMA. PGA. panel. win)
      else
      ETL. TX2 = Call MERGE-BR-LOCATION-STRUCTURE-PROC (the symbol  motif. type0. type1. type7.
        PETL. PMA. panel. win):
  else
  if PGA exists then
    if PGA-2 exists then
      if there exists a scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type9]-[type10]-ETL") then
        go to final-routine
      else
        ETL. TX2 = call MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (the symbol group.
          the symbol group. type0. type1. type9. type10. PETL. PGA. PGA-2. panel. win)
      else
      ETL. TX2 =Call MERGE-BR-LOCATION-STRUCTURE-PROC (the symbol group. type0. type1. type9.
        PETL. PGA. panel. win):
  return ETL. TX2:
final-routine:
  return ETL. TX2:
```

TABLE 164

```
MERGE-BR-LOCATION-STRUCTURE-PROC (tag1: symbol. type0): symbol. type1: symbol. type3: symbol.
  PETL: class scroll-text-list. QA: class query-array. panel: class query-panel. win : class window) =
  (class scroll-text-list. text)

PN: symbol = the name of panel:
BRN: symbol = the related-item of panel:
  if there exists a e-scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type3]-ETL") then
    go to exit-routine
  else
    create a e-scroll-text-list ETL:
    conclude that the label of ETL = "[PN]-[type0]-[type1]-[type3]-ETL":
    {conclude that ETL is a-list-associated-to-dlg panel:}
    transfer ETL to the subworkspace of safe-home-for-lists at (50. -245):
    {TX2 = the scroll-title of ETL:}
    if the number of elements in PETL = 0 then TX2 = capitalize-words ("There are no structures  defined in this
      module that are [type0] of [BRN] and located in the [type1]") :
  else
    if the array-length of QA = 0 then
      if type3 = symbol ("no-[tag1]-selection") then
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
          located in the [type1]")
      else
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
          contained in the [type1] .  None of those contain [type3]");
      for T1 = each text in PETL do
        insert T1 at the end of the text list ETL:
    else
      TX2 = capitalize-words ("Entities that are [type0] of [BRN] and located in the [type1] that contain [type3]");
      for T1 = each text in PETL do
        for S2 = each symbol in QA do
          if T1 = "[S2]" then insert T1 at the end of the text list ETL:
  return ETL. TX2:
exit-routine:
    TX2 = the scroll-title of ETL:
    return ETL. TX2:
```

TABLE 165

```
MERGE-BR-LOCATION-STRUCTURE-STRUCTURE-PROC (tag1: symbol, tag2: symbol, type0: symbol, type1:
symbol, type4: symbol, type5: symbol, PETL: class scroll-text-list, QA1: class query-array, QA2: class query-array,
panel: class query-panel, win : class window) = (class scroll-text-list, text)
BRN: symbol = symbol (the related-item of panel):
PN: symbol = the name of panel:
 if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type0]-[type1]-[type4]-[type5]-ETL") then
  go to exit-routine
 else
   create a c-scroll-text-list ETL:
   conclude that the label of ETL = "[PN]-[type0]-[type1]-[type4]-[type5]-ETL":
   conclude that ETL is a-list-associated-to-dlg panel:
   transfer ETL to the subworkspace of safe-home-for-lists at (50, -245):
   TX2 = the scroll-title of ETL:
   if the array-length of QA1 = 0 then
     if the array-length of QA2 = 0 then
       if type5 = symbol ("no-[tag2]-selection") and type4 = symbol ("no-[tag1]-selection") then
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1]")
       else
        if type5 = symbol ("no-[tag2]-selection") then
          TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
             located in the [type1], None contain [type4]")
       else
        if type4 = symbol ("no-[tag1]-selection") then
          TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and l
             located in the [type1] contain [type5]")
       else
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
             located in the [type1]"):
      for T = each text in PETL do
       insert T at the end of the text list ETL:
     else if the array-length of QA2 >= 0 then
       if type4 = symbol ("no-[tag1]-selection") then
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1] that contain [type5]")
       else
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1] that contain [type5]. None was found also containing [type4]"):
        for T = each text in PETL do
         for S2 = each symbol in QA2 do
          if T = "[S2]" then
           insert T at the end of the text list ETL:
   else if the array-length of QA1 >= 0 then
     if the array-length of QA2 = 0 then
      if type5 = symbol ("no-[tag2]-selection") then
       TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1] that contain [type4]")
       else
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1] that contain [type4]. None was found also containing [type5]"):
        for T = each text in PETL do
         for S1 = each symbol in QA1 do
          if T = "[S1]" then
           insert T at the end of the text list ETL:
      else if the array-length of QA2 >= 0 then
        TX2 = capitalize-words ("Entities with structures defined in this module that are [type0] of [BRN] and
           located in the [type1] that contain [type4] and [type5]"):
        for T = each text in PETL do
         for S1 = each symbol in QA1 do
          if T = "[S1]" then
           for S2 = each symbol in QA2 do
            if T = "[S2]" then
             insert T at the end of the text list ETL:
 return ETL, TX2:
exit-routine:
  TX2 = the scroll-title of ETL:
  return ETL, TX2:
```

TABLE 166

```
QUERY-BR-FUNCTION-STRUCTURE-LISTS-PROC (type0: symbol, type2: symbol, PETL: class scroll-text-list,
panel: class query-panel, win : class window) = (class scroll-text-list, text)
PN: symbol = the name of panel;
SW: class kb-workspace = the subworkspace of panel;
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  type3, PSA = call FIND-SPECIFIC-ARRAY-PROC ("subunits", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  type4, PSA-2 = call FIND-SPECIFIC-ARRAY-PROC ("subunits-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  type5, PDA = call FIND-SPECIFIC-ARRAY-PROC ("domains", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  type6, PDA-2 = call FIND-SPECIFIC-ARRAY-PROC ("domains-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  type7, PMA = call FIND-SPECIFIC-ARRAY-PROC ("motifs", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  type8, PMA-2 = call FIND-SPECIFIC-ARRAY-PROC ("motifs-2", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  type9, PGA = call FIND-SPECIFIC-ARRAY-PROC ("groups", panel, win);
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  type10, PGA-2 = call FIND-SPECIFIC-ARRAY-PROC ("groups-2", panel, win);
 if PSA exists then
    if PSA-2 exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type3]-[type4]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol subunit, the
symbol subunit, type0, type2, type3, type4, PETL, PSA, PSA-2, ETL, panel, win);
    else
      ETL, TX2  = call C-MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol subunit, type0, type2, type3,
PETL, PSA, ETL, panel, win);
 else
 if PDA exists then
    if PDA-2 exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type5]-[type6]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain, the
symbol domain, type0, type2, type5, type6, PETL, PDA, PDA-2, ETL, panel, win)
    else
    if PMA exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type5]-[type7]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain, the
symbol motif, type0, type2, type5, type7, PETL, PDA, PMA, ETL, panel, win)
    else
    if PGA exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type5]-[type9]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol domain, the
symbol group, type0, type2, type5, type9, PETL, PDA, PGA, ETL, panel, win)
    else
      ETL, TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-PROC  (the symbol domain, type0, type2, type5,
PETL, PDA, ETL, panel, win);
 else
 if PMA exists then
    if PMA-2 exists then
       if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type7]-[type8]-ETL") then
         go to final-routine
       else
         ETL, TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol motif, the
symbol motif, type0, type2, type7, type8, PETL, PMA, PMA-2, ETL, panel, win.
    else
```

continue

TABLE 166 cont.

```
  if PGA exists then
    if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type7]-[type9]-ETL") then
      go to final-routine
      ETL. TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol motif.
        the symbol group. type0. type2. type7. type9. PETL. PMA. PGA. ETL. panel. win)
  else
    ETL. TX2 = Call C-MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol motif. ype0. type2. type7.
      PETL. PMA. ETL. panel. win):
else
 if PGA exists then
   if PGA-2 exists then
     if there exists a c-scroll-text-list ETL such that (the label of ETL = "[PN]-[type9]-[type10]-ETL") then
      go to final-routine
     else
       ETL. TX2 = call C-MERGE-BR-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol group.
         the symbol group. type0. type2. type9. type10. PETL. PGA. PGA-2. ETL. panel. win)
   else
     ETL. TX2 =Call C-MERGE-BR-FUNCTION-STRUCTURE-PROC (the symbol  group. type0. type2.
       type9. PETL. PGA. ETL. panel. win):
 return ETL. TX2:
final-routine:
  return ETL. TX2:
```

TABLE 167

```
QUERY-BR-LOCATION-FUNCTION-STRUCTURE-LISTS-PROC (type0: symbol. type1: symbol. type2:
symbol. PETL: class scroll-text-list. panel: class query-panel. win : class window) = (class scroll-text-list. text)
BRN: symbol = symbol (the related-item of panel):
PN: symbol = the name of panel:
SW: class kb-workspace = the subworkspace of panel:
 if the number of elements in PETL = 0 then
   TX2 = capitalize-words ("There are no entities with structures defined in this module that are [type0] of [BRN]
     and located in the [type1]")
else
 if there exists a radio-button B upon SW such that (the id of B = "subunits") then
  type3. PSA = call FIND-SPECIFIC-ARRAY-PROC ("subunits". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "subunits-2") then
  type4. PSA-2 = call FIND-SPECIFIC-ARRAY-PROC ("subunits-2". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "domains") then
  type5. PDA = call FIND-SPECIFIC-ARRAY-PROC ("domains". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "domains-2") then
  type6. PDA-2 = call FIND-SPECIFIC-ARRAY-PROC ("domains-2". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "motifs") then
  type7. PMA = call FIND-SPECIFIC-ARRAY-PROC ("motifs". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "motifs-2") then
  type8. PMA-2 = call FIND-SPECIFIC-ARRAY-PROC ("motifs-2". panel. win):
  else
 if there exists a radio-button B upon SW such that (the id of B = "groups") then
  type9. PGA = call FIND-SPECIFIC-ARRAY-PROC ("groups". panel. win)
  else
 if there exists a radio-button B upon SW such that (the id of B = "groups-2") then
  type10. PGA-2 = call FIND-SPECIFIC-ARRAY-PROC ("groups-2". panel. win):
 if PSA exists then
   if PSA-2 exists then
     if there exists a scroll-text-list ETL such that (the label of ETL =
       "[PN]-[type0]-[type1]-[type2]-[type3]-[type4]-ETL") then
      go to final-routine
     else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
         subunit. the symbol subunit. type0. type1. type2. type3. type4. PETL. PSA. PSA-2. panel. win):
   else
     ETL. TX2  = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-PROC the symbol subunit.
       (type0. type1. type2. type3. PETL. PSA. panel. win):
```

continue

TABLE 167 cont.

```
else
if PDA exists then
   if PDA-2 exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type5]-[type6]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          domain. the symbol domain. type0. type1. type2. type5. type6. PETL. PDA. PDA-2. panel. win)
   else
   if PMA exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type5]-[type7]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          domain. the symbol motif. type0. type1. type2. type5. type7. PETL. PDA. PMA. panel. win)
   else
   if PGA exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type5]-[type9]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          domain. the symbol group. type0. type1. type2. type5. type9. PETL. PDA. PGA. panel. win)
   else
    ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-PROC (the symbol domain.
       type0. type1. type2. type5. PETL. PDA. panel. win):
 end
else
if PMA exists then
   if PMA-2 exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type7]-[type8]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          motif. the symbol motif. type0. type1. type2. type7. type8. PETL. PMA. PMA-2. panel. win)
   else
   if PGA exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type7]-[type9]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          motif. the symbol group. type0. type1. type2. type7. type9. PETL. PMA. PGA. panel. win)
   else
    ETL. TX2 = Call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-PROC (the symbol motif. type0.
       type1. type2. type7. PETL. PMA. panel. win):
else
if PGA exists then
   if PGA-2 exists then
      if there exists a scroll-text-list ETL such that (the label of ETL =
        "[PN]-[type0]-[type1]-[type2]-[type9]-[type10]-ETL") then
       go to final-routine
      else
       ETL. TX2 = call MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (the symbol
          group. the symbol group. type0. type1. type2. type9. type10. PETL. PGA. PGA-2. panel. win)
   else
    ETL. TX2 =Call C-MERGE-BR-LOCATION-FUNCTION-STRUCTURE-PROC (the symbol group. type0.
       type1. type2. type9. PETL. PGA. panel. win):
return ETL. TX2:
final-routine:
  TX2 = the scroll-title of ETL:
  return ETL. TX2:
```

TABLE 11

Class name	rate-pvar
Superior class	float-pvar
Attributes specific to class	none
Default settings	initial value for float-parameter: 0.0: history keeping spec: keep history with maximum age of data points = 10 minutes
Class name	binding-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	tranlocation-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	activation-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	interaction-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	cell-progression-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	produc-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	consum-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	output-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none
Class name	input-rate-pvar
Superior class	rate-pvar
Attributes specific to class	none
Default settings	none

TABLE 12

Class name	velocity-pvar
Superior class	float-pvar
Attributes specific to class	none
Menu option	not a final menu choice
Default settings	none
Class name	enzyme-velocity
Superior class	velocity-pvar
Attributes specific to class	none
Menu option	not a final menu choice
Default settings	none
Class name	receptor-velocity
Superior class	velocity-var
Attributes specific to class	none
Class name	complex-formation-velocity
Superior class	velocity-pvar
Attributes specific to class	none
Class name	complex-dissoc-velocity
Superior class	velocity-pvar
Attributes specific to class	none
Class name	ion-transport-velocity
Superior class	velocity-pvar
Attributes specific to class	none
Class name	conform-change-velocity
Superior class	velocity-pvar
Attributes specific to class	none
Class name	translocation-velocity
Superior class	velocity-pvar
Attributes specific to class	none

TABLE 13

Class name	float-var
Superior class	float-variable
Attributes specific to class	none
Inherited attributes	none
Default settings	options for variable: do forward chain, do not backward chain: data server: G2 simulator: validity interval: indefinite: supply simulation-subtable
Stubs	inherited
Class name	velocity-var
Superior class	float-var
Attributes specific to class	none
Menu option	a final menu choice
Default settings	none
Class name	concentration-var
Superior class	float-var
Attributes specific to class	none
Menu option	a final menu choice
Default settings	none
Class name	density-var
Superior class	float-var
Attributes specific to class	none
Menu option	a final menu choice
Default settings	none
Class name	kinetic-parameter-var
Superior class	float-var
Attributes specific to class	none
Menu option	a final menu choice
Default settings	initial value for float-variable: 9.9e-99

TABLE 14

Class name	model-block-var
Superior class	float-var
Attributes specific to class	none
Default settings	default update interval: 3 seconds supply simulation subtable:
Class name	model-block-output-var
Superior class	float-var
Attributes specific to class	none
Default settings	validity interval: indefinite: history keeping spec: keep history with maximum age of data points = 20 minutes: initial value for float-variable: 1.0e-19: default update interval: 3 seconds
Class name	graph-transf-var
Superior class	float-var
Attributes specific to class	none
Default settings	initial value for float-variable: 0.0: default update interval: 3 seconds: history keeping spec: keep history with maximum age of data points = 10 minutes

TABLE 15

Class name	transf-factor-var
Superior class	float-variable
Attributes specific to class	transform is given by a graph-transf-var: ref-variable: label: x is x
Inherited attributes	none
Default settings	initial value for float-variable: 0.0: default update interval: 3 seconds: history keeping spec: keep history with maximum number of data points = 3
Attribute displays	label offset by (-148, 7): x offset by (-20, 7) last-recorded-value offset by (-10, 6):
Icon description	transf-factor-var-pattern (width 320; height 24; edge-color = foreground)
Class name	velocity-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is scaled-velocity
Class name	input-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is input-rate
Class name	output-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is output-rate
Class name	accum-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is accumulation-rate
Class name	scaled-amount-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is scaled-conc
Class name	density-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is density
Class name	conc-transf-factor
Superior class	transf-factor-var
Attributes specific to class	label is concentration

TABLE 16

Class name	query-array
Superior class	symbol-array
Attributes specific to class	major-class: ref-component-class
Default settings	initial values for symbol-array: none
Attribute displays	ref-component-class at (-10, -15)
Icon description	query-array-pattern (width 38; height 38; medium-orchid, cyan)

a query-array	
Names	none
Array lenght	0
Element type	symbol
Initial values	none
Major class	none
Ref component class	none

TABLE 17

Class name	references-array
Superior class	text-array
Attributes specific to class	none
Default settings	initial values for symbol-array: none
Attribute displays	ref-component-class at (-10, -15)
Icon description	references-array-pattern (width 38; height 38; violet, cyan)

TABLE 18

Class name	query-list
Superior class	symbol-list
Attributes specific to class	lenght is 0: ref-array
Default settings	allow duplicate elements for g2-list: no
Attribute displays	ref-array at (-10, -15)
Icon description	query-list-pattern (width 36; height 40; salmon; medium-blue)

a query-list	
Names	none
Element type	symbol
Allow duplicate elements?	no
Lenght	0
Ref array	none

TABLE 19

Class name	scroll-text-list
Superior class	text-list
Attributes specific to class	label is scroll-list
Class restrictions	unless in administrator or developer mode: selecting any c-scroll-text-list implies describe
Default settings	allow duplicate elements for g2-list: no
Attribute displays	label offset by (-32, -25)

TABLE 20

Class name	bioitem-list
Superior class	item-list
Attributes specific to class	none
Class restrictions	unless in administrator or developer mode: selecting any bioitem-list implies describe
Icon description	bioitem-list-pattern (width 36: height 40)
Class name	experiment-selections-list
Superior class	bioitem-list
Attributes specific to class	label is "Exper. Selections"
Default settings	allow duplicate elements for g2-list: no: element type for item-list: experiment-selection
Attribute displays	label offset by (-65, -26)
Icon description	experiment-selections-list-pattern (width 36: height 40)
Class name	bioreservoirs-list
Superior class	bioitem-list
Attributes specific to class	list-is-completed has values true or false (default is FALSE): label
Default settings	element type for item-list: bioreservoir: allow duplicate elements for g2-list: no
Icon description	bioreservoirs-list-pattern (width 36: height 40)
Class name	downstream-bioreservoirs-list
Superior class	bioreservoirs-list
Attributes specific to class	label is "DBRL"
Attribute displays	label offset by (-26, -25)
Class name	bioprocesses-list
Superior class	bioitem-list
Attributes specific to class	list-is-completed has values true or false (default is FALSE): label
Default settings	element type for item-list: bioprocess: allow duplicate elements for g2-list: no
Icon description	inherited
Class name	downstream-bp-simulation-list
Superior class	bioprocesses-list
Attributes specific to class	label is "SDBPL"
Attribute displays	label offset by (-30, -25)
Class name	bm-downstream-bp-simulation-list
Superior class	downstream-bp-simulation-list
Attributes specific to class	ref-biomodel
Inherited attributes	label is "SDBPL"
Attribute displays	inherited
Class name	neural-inputs-list
Superior class	bioitem-list
Attributes specific to class	label is "N.In"
Attribute displays	label offset by (-25, -26)
Icon description	neural-inputs--pattern (width 36: height 40)
Class name	neural-outputs-list
Superior class	bioitem-list
Attributes specific to class	label is "N.Out"
Attribute displays	label offset by (-25, -26)
Icon description	neural-outputs-list-pattern (width 36: height 40)
Class name	patheross-bp-list
Superior class	bioprocesses-list
Attributes specific to class	label is "XBPL"
Attribute displays	label offset by (-26, -25)
Icon description	patheross-bp-pattern (width 36: height 40)
Class name	patheross-br-list
Superior class	bioreservoirs-list
Attributes specific to class	label is "XBRL"
Attribute displays	label offset by (-26, -25)
Icon description	patheross-br-list-pattern (width 36: height 40)

TABLE 21

Class name	biotool
Superior class	object
Attributes specific to class	none
Menu option	not a final menu choice

TABLE 22

Class name	model-block
Superior class	biotool
Attributes specific to class	output-1 is given by a model-block-output-var
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: move, name, change-size, color, describe, create-subworkspace; when in simulation, navigation, or general mode: menu choices exclude additionally: clone, delete; attributes visible exclude additionally: notes, names
Menu option	not a final menu choice
Icon description	model-block-pattern
Class name	proportional.f
Superior class	model-block
Attributes specific to class	input-1 is given by a model-block-var; gain is 1.0; bias is 0.0
Class name	sigmoid.f
Superior class	model-block
Attributes specific to class	input-1; gain is 1.5; bias is 0.0
Class name	exp.c.pdf
Superior class	model-block
Attributes specific to class	mean is 0.0; x-val is given by a model-block-var
Class name	unif.c.pdf
Superior class	model-block
Attributes specific to class	min-val is 1.0e-5; max-val is 1.0; x-val is given by a model-block-var
Class name	unif.m.f
Superior class	model-block
Attributes specific to class	min-val is 1.0e-5; max-val is 1.0
Class name	normal.i.pdf
Superior class	model-block
Attributes specific to class	mean is 0.0; min-val is 1.0e-5; max-val is 1.0; x-val
Class name	sin.time.f
Superior class	model-block
Attributes specific to class	min-val is 1.0e-5; max-val is 1.0; frequency is 25; gain is 2; bias is 1.0
Class name	generic.model.block
Superior class	model-block
Attributes specific to class	p.1; p.2; p.3; p.4; p.5; p.6; p.7; p.8; p.9; v.1; v.2; v.3; v.4; v.5; v.6; v.7; v.8; v.9
Icon description	generic.model.block-pattern (omitted)

TABLE 23

Class name	inference-block
Superior class	biotool
Attributes specific to class	label is ""; description is ""; outcome is given by an outcome-par; beta-coeff is given by a beta-coeff-par
Class restrictions	unless in administrator or uil-build mode: menu choices exclude additionally: move. name. change-size. color. describe. create-subworkspace. rotate-reflect; when in simulation. navigation. or explorer mode: menu choices exclude additionally: clone. delete; attributes visible exclude additionally: notes. names. user-restrictions
Menu option	not a final menu choice
Attribute displays	label offset by (-35. 45)
Stubs	a p-connection located at right 26
Icon description	width 52; height 52; foreground

Class name	deactivation-block
Superior class	inference-block
Attributes specific to class	none
Menu option	not a final menu choice

Class name	time-deactivation-block
Superior class	deactivation-block
Attributes specific to class	time-interval is given by a deactivation-interval-par; time-threshold is 9.9e99

Class name	event-deactivation-block
Superior class	deactivation-block
Attributes specific to class	none

Class name	activation-block
Superior class	inference-block
Attributes specific to class	none
Icon description	inherited

Class name	time-activation-block
Superior class	activation-block
Attributes specific to class	time-interval is given by an activation-interval-par; time-threshold is 9.9e99
Icon description	inherited

Class name	checkpoint-to-floor-block
Superior class	activation-block
Attributes specific to class	checkpoint is given by a checkpoint-var; threshold is 9.9e99
Icon description	inherited

Class name	checkpoint-to-ceiling-block
Superior class	activation-block
Attributes specific to class	checkpoint is given by a checkpoint-var; threshold is 9.9e99
Icon description	inherited

a checkpoint-to-ceiling-block

Names	none
Label	""
Description	""
Outcome	within-threshold
Beta coeff	0.0
Checkpoint	****
Threshold	9.9e99

TABLE 24

Class name	button
Superior class	biotool
Attributes specific to class	toggle-state has values show or hide (default is HIDE)
Icon description	button-pattern (width 22; height 22; dark-color = dim-gray, light-color = light-gray)
Class name	action-button
Superior class	button
Attributes specific to class	label: state has values enabled or disabled: action-proc-name:
Class restrictions	unless in administrator mode: selecting any action-button implies select: unless in administrator, developer, or modeler mode: non-menu choices exclude additionally: move-object
Class name	hide-sup-ws-button
Superior class	action-button
Attributes specific to class	action-proc-name is HIDE-SUP-WS-CALLBACK
Icon description	hide-sup-ws-button-pattern

TABLE 25

an user-menu-choice

Label	select
Applicable class	action-button
Condition	the state of the item is enabled
Action	start BUTTON-HANDLER-CALLBACK (the item, this window)

```
BUTTON-HANDLER-CALLBACK (button: class action-button, win : class window)
  start the procedure named by the action-proc-name of button (button, win):
  change the color-pattern of button so that light-color is light-gray,
    dark-color is dim-gray :
  conclude that the toggle-state of button is show:
  if button has a face-color FC then change FC to gray
```

TABLE 26

```
HIDE-SUP-WS-CALLBACK (button: class action-button, win: class window)
  WS = the workspace of button:
  OBJ = the object superior to WS:
  change the size of WS to minimum:
  hide WS on win:
  if the user-mode of win is not simulation then:
    if OBJ is a bioprocess then:
      change the status-color of OBJ to slate-blue
    else if OBJ is a bioreservoir then:
      change the status-color of OBJ to medium-aquamarine:
  if the toggle-state of OBJ is show then conclude that the
    toggle-state of OBJ is hide:
```

TABLE 27

Class name	change-mode-button
Superior class	action-button
Attributes specific to class	action-proc-name is CHANGE-MODE-CALLBACK: mode has values administrator, developer, modeler, explorer, simulation, default is MODELER
Icon description	change-mode-button-pattern

```
CHANGE-MODE-CALLBACK (button: class change-mode-button, win: class window)
  mode = the mode of button:
  call CHANGE-MODE-PROC (mode, win):
  for each change-mode-button B upon the workspace of button do
    change the color-pattern of B so that face is forest-green:
    change the color-pattern of button so that face is green:
  return:
```

```
CHANGE-MODE-PROC (mode: symbol, win : class window)
 if the text of the g2-user-mode of win /= "simulation" and mode is simulation then
    inform the operator that "The subworkspaces of all bioReservoirs and all
      bioProcesses will be deactivated":
    for BR = each bioreservoir do
     allow other processing:
     if the subworkspace of BR exists then
        deactivate the subworkspace of BR:
        change the status-color icon-color of BR to aquamarine :
    inform the operator that "The subworkspaces of all bioReservoirs have been
      deactivated":
    for BP = each bioprocess do
     allow other processing:
     if the subworkspace of BP exists then
        deactivate the subworkspace of BP:
        change the status-color icon-color of BP to aquamarine:
    inform the operator that "The subworkspaces of all bioProcesses have been
      deactivated":
 else
  if the text of the g2-user-mode of win = "simulation" and mode is not simulation then
    inform the operator that "The subworkspaces of all
      bioReservoirs and all bioProcesses will be activated":
    for BR = each bioreservoir do
     allow other processing:
     if the subworkspace of BR exists then activate the subworkspace of BR:
    inform the operator that "The subworkspaces of all bioReservoirs have been
      activated":
    for BP = each bioprocess do
     allow other processing:
     if the subworkspace of BP exists then activate the subworkspace of BP:
    inform the operator that "The subworkspaces of all bioProcesses have been
      activated":
 case (mode) of
  administrator:
    change the text of the g2-user-mode of win to "administrator":
  developer:
    change the text of the g2-user-mode of win to "developer":
  modeler:
    change the text of the g2-user-mode of win to "modeler":
  general :
    change the text of the g2-user-mode of win to "general":
  navigation :
    change the text of the g2-user-mode of win to "navigation":
  simulation:
    change the text of the g2-user-mode of win to "simulation"
```

TABLE 28

Class name	path-tracer
Superior class	button
Attributes specific to class	none
Class restrictions	unless in administrator or developer mode: selecting any path-tracer implies show
Menu option	not a final menu choice
Icon description	path-tracer-pattern (width 14: height 21)

Class name	down-path-tracer
Superior class	path-tracer
Attributes specific to class	none
Icon description	down-path-tracer-pattern

Class name	up-path-tracer
Superior class	path-tracer
Attributes specific to class	none
Icon description	up-path-tracer-pattern (width 14: height 20)

Class name	navig-path-tracer
Superior class	path-tracer
Attributes specific to class	none
Icon description	up-path-tracer-pattern (width 14: height 20)

TABLE 29

an user-menu-choice

Label	show
Applicable class	path-tracer
Condition	none
Action	start PATH-TRACER-HANDLER-PROC (the item, this window)

```
PATH-TRACER-HANDLER-PROC (tracer: class path-tracer, win:
    class window)
  if the toggle-state of tracer is hide then
    if the subworkspace of tracer exists then
      case the class of tracer is
        up-path-tracer:
          call DRAW-UP-PATHWAY-PROC (tracer, win);
        down-path-tracer :
          call DRAW-DOWN-PATHWAY-PROC (tracer, win);
        complex-down-path-tracer :
          call DRAW-COMPLEX-DOWN-PATHWAY-PROC (tracer, win);
  if the toggle-state of tracer is show then
    if the SW of tracer exists then hide the SW of tracer;
  call CONFIGURE-TRACER-PROC (tracer, win);
  return
```

```
CONFIGURE-TRACER-PROC (tracer: class tracer, win: class window)
  case the toggle-state of tracer is
    hide:
      change the color-pattern of tracer so that light-color is dim-gray,
        dark-color is light-gray , white-color = the symbol black;
      conclude that the toggle-state of tracer is show;
    show:
      change the color-pattern of tracer so that light-color is light-gray,
        dark-color is dim-gray , white-color = the symbol white;
      conclude that the toggle-state of tracer is hide;
  return
```

TABLE30

Class name	lists-tracer
Superior class	button
Attributes specific to class	none
Class restrictions	unless in administrator or developer mode: selecting any lists-tracer implies show: non-menu choices exclude additionally: move-object:
Menu option	not a final menu choice
Icon description	lists-tracer-pattern (width 14: height 21)

Class name	bp-chaininig-tracer
Superior class	lists-tracer
Attributes specific to class	none
Icon description	bp-chaininig-tracer-pattern

Class name	br-chaininig-tracer
Superior class	lists-tracer
Attributes specific to class	none
Icon description	br-chaininig-tracer-pattern (width 14: height 20)

Class name	basal-density-source-tracer
Superior class	lists-tracer
Attributes specific to class	none
Icon description	basal-density-source-tracer-pattern (width 24: height 24)

Class name	br-scroll-tracer
Superior class	lists-tracer
Attributes specific to class	none
Icon description	br-chaininig-tracer-pattern (width 116: height 30)

Class name	query-tracer
Superior class	lists-tracer
Attributes specific to class	none
Class restrictions	unless in administrator or developer mode: selecting any query-tracer implies show: non-menu choices exclude additionally: move-object:
Icon description	lists-tracer-pattern (width 14: height 21)

TABLE31

an user-menu-choice

Label	show
Applicable class	lists-tracer
Condition	none
Action	start LIST-TRACER-HANDLER-PROC (the item. this window)

```
LIST-TRACER-HANDLER-PROC (tracer: class tracer. win: class window)
  if the toggle-state of tracer is hide then
    casethe class of tracer is:
      bp-chaininig-tracer:
        call BP-CHAINING-PROC (the bioprocess superior to the WS of tracer. win):
      br-chaininig-tracer :
        call BR-CHAINING-PROC (the bioreservoir superior to the WS of tracer. win):
      br-scroll-tracer:
        call BR-SCROLL-PROC (tracer. win):
  else if the toggle-state of tracer is show and the SW of tracer exists then
    delete the subworkspace of tracer:
  call CONFIGURE-TRACER-PROC (tracer. win):
  return
```

TABLE 32

Class name	graph-tracer
Superior class	button
Attributes specific to class	none
Class restrictions	when in simulation or developer mode: selecting any graph-tracer implies show-plot:
Menu option	not a final menu choice
Icon description	graph-tracer-pattern (width 14: height 20)
Class name	scaled-br-graph-tracer
Superior class	graph-tracer
Attributes specific to class	none
Icon description	scaled-br-graph-tracer-pattern (width 14: height 20)
Class name	absolute-br-graph-tracer
Superior class	graph-tracer
Attributes specific to class	none
Icon description	absolute-br-graph-tracer-pattern (width 14: height 21)
Class name	scaled-bp-graph-tracer
Superior class	graph-tracer
Attributes specific to class	none
Icon description	scaled-bp-graph-tracer-pattern (width 14: height 20)

TABLE 33

an user-menu-choice	
Label	show-plot
Applicable class	graph-tracer
Condition	the user-mode of this window is simulation or the user-mode of this window is developer
Action	start TRANSF-GRAPH-TRACER-HANDLER-PROC (the item. this window)

```
TRANSF-GRAPH-TRACER-HANDLER-PROC (tracer: class graph-
tracer. win: class window )
  if the toggle-state of tracer is hide then:
    if the SW of tracer exists then
      make the SW of tracer transient:
      delete the SW of tracer:
    case the class of tracer is
      scaled-br-graph-tracer:
        call SCALED-BR-TRANSF-GRAPH-PROC (tracer. win):
      scaled-bp-graph-tracer:
        call SCALED-BP-TRANSF-GRAPH-PROC (tracer. win):
      absolute-br-graph-tracer:
        call ABSOLUTE-BR-TRANSF-GRAPH-PROC (tracer. win):
  else if the toggle-state of tracer is show then
    delete the SW of tracer:
  call CONFIGURE-TRACER-PROC (tracer. win):
  return
```

TABLE 34

Class name	reference-block
Superior class	biotool
Attributes specific to class	id. with an index: short-ref is "". with an index: unique-identifier. with an index: author-s is "Smith AB. Morris AD. title is "": institution is "Dept. Institution. Country": journal is "Journal Full Name Ref": abstract is ""
Class restrictions	unless in administrator. developer. or modeler mode: menu choices exclude additionally: transfer. name. rotate-reflect. change-size. disable. describe: attributes visible for cabe-reference exclude additionally: user-restrictions. names. id: selecting any cabe-reference implies table
Attribute displays	short-ref offset by (8. 14)
Icon description	reference-pattern (width 27: height 35

TABLE 35

Class name	bio-object-title
Superior class	biotool
Attributes specific to class	label is "long.name.of.biomodel"
Class restrictions	unless in administrator or developer mode: menu choices exclude additionally: delete. transfer. name. disable. describe. create-subworkspace
Attribute displays	label offset by (-72. -7)
Icon description	bio-object-title-pattern (width 130: height 2)

TABLE 36

MESSAGE DEFINITIONS

Class name	cabe-reference-message
Superior class	message
Attributes specific to class	none
Default message properties	none

Class name	sequence-display
Superior class	message
Attributes specific to class	id. with an index: label is ""
Default message properties	minimum-with is 100: text-color is transparent: background color is foreground

TABLE 37

Class name	entry-panel
Superior class	uil-tailored-dialog
Attributes specific to class	id is "". with an index: related-item is none: current-simulation-time is given by an elapsed-simul-time-par: current-simulation-time-increment is given by an elapsed-simul-time-par
Menu option	a final menu choice
Inherited attributes	several proprietary attributes
Icon description	entry-panel (width 80: height 100: dark-color = dim-gray. light-color = light-gray. white-color = white)

Class name	query-panel
Superior class	uil-tailored-dialog
Attributes specific to class	id is "". with an index: related-item is none
Inherited attributes	several proprietary attributes
Icon description	entry-panel (width 80: height 100: dark-color = dim-gray. light-color = light-gray. white-color = white)

TABLE 38

CONNECTION DEFINITIONS

Class name	p-connection
Superior class	connection
Attributes specific to class	none
Inherited attributes	none
Cross section pattern	core = dim-gray. cover = green:1 cover. 2 core. 1 cover
Stub length	8
Class name	r-connection
Superior class	connection
Attributes specific to class	none
Inherited attributes	none
Cross section pattern	core = red. cover = dim-gray: 2 cover. 2 core. 2 cover
Stub length	8
Class name	model-box-connection
Superior class	connection
Attributes specific to class	none
Inherited attributes	none
Cross section pattern	core = red. cover = green: 2 cover. 2 core
Stub length	8
Class name	cycle-path
Superior class	connection
Attributes specific to class	tau-coeff is given by a tau-coeff-par: rate-constant is given by a cell-progression-rate-constant-par: progression-rate is given by a cell-progression-rate-pvar
Cross section pattern	core = gray. cover = green: 2 cover. 6 core. 2 cover
Stub length	10

TABLE 39

Class name	biopost
Superior class	connection-post
Attributes specific to class	toggle-state has values show or hide (default is HIDE)
Class restrictions	unless in administrator or developer mode: attributes visible exclude additionally: superior-connection. toggle-state
Menu option	not a final menu choice

TABLE 40

Class name	bio-object
Superior class	object
Attributes specific to class	label is "". with an index: description is ""
Class restrictions	unless in administrator. developer. or modeler mode: menu-choices exclude additionally: move-object. rotate-reflect
Menu option	not a final menu choice
Icon description	bio-object-pattern
Class name	bionode-object
Superior class	bio-object
Attributes specific to class	none
Menu option	not a final menu choice
Icon description	inherited
Class name	bioview-object
Superior class	bio-object
Attributes specific to class	references is "": warnings is "": toggle-state has values show or hide (default is HIDE)
Menu option	not a final menu choice
Icon description	inherited

TABLE 41a

+ BioEntity
I. Simple-bioEntity
A. Protein-site
Ligand.site
...
Catalytic.site
...
Tyr.site
Ser.site
Thr.site
Cys.site
...
Isoprenylation-site
Myristoylation-site
...
Protein-loss-of-function-mutation
Protein-gain-of-function-mutation
...
B. Protein-modified-group
Tyr.P-group
Ser.P-group
Thr.P-group
Disulphide-bridge
Isoprenylated-group
Myristoylated-group
...
C. ...
D. DNA-binding-site
Transcription-element
...
Enhancer-element
...
Promoter-proximal-element
...
Response-element
...
Activator-site
...
...
E. DNA-modification-site
Methylation.site
...
F. RNA-binding-site
...
G. ...
II. Complex-bioEntity
A. BioEntity-Component
1. Protein-Component
a. Protein-domain
Cytoplasmic-domain
Transmembrane-domain
Extracellular-domain
COOH-terminal-domain
NH2-terminal-domain
Protein-spacer
SH2-domain
SH3-domain
HSP-domain
Alpha-ß-domain
ß-sheet-domain
Helix-turn-helix-domain
Ig-fold-domain
Ig-CH1-domain
...
Collagen-domain
Fibronectin-type-III-domain
Superbarrel-domain
8-stranded-barrel-domain
...
b. Protein-motif
Leucin-zipper-motif
Zinc-finger-motif
Alpha-helix-motif
Helix-turn-motif
ß-strand-motif
EF-hand-motif
Trefoil-motif
Greek-key-motif
Loop-region-motif
Signal-peptide-motif
...
c. ...
2. DNA-Component
a. DNA-domain
Operator
...
Promoter
...
b. DNA-motif
TATA-box
...
c. RNA-domain
...
d. RNA-motif
...
3. Membrane
Plasma.Membrane
Nuclear.Membrane
...
...
B. Molecule
1. Protein
a. Active.polypeptide
Cytokine
EPO.like
...
IL2.like
...
IL4.like
...
IL6.like
...
IL7.like
...
LIF.like
...
...
Growth-factor
EGF.like
...
PDGF.like
...
...
Peptide.hormone
GH.like
...
Insulin.like
...
...
b. Receptor
PTK.Rcp
EGF.R.like
EGF.R
HER-2/neu
erbB3
...
PDGF.R.like
...
Insulin.R.like
...
FGF.R.like
...
...
PSTK.Rcp
...
PPase.Rcp
...
Hematopoietin.R.like
...
Ig.R.like
IL1.R.like
...
G.coupled.R.like
IL8.R.like
...
...
Steroid-receptors
...
...
...
Receptor -complex
PTK.Rcp.complex
EGF.R.like.complex
EGF-EGF.R
EGF-HER-2/neu
EGF-erbB3
...
PDGF.R.like.complex
PDGF-PDGF.R
CSF1-CSF1.R/fms
...
Insulin.R.likecomplex
Insulin-Insulin.R
IGF-IGF1.I
...

TABLE 1b

FGF.R.like.complex
FGF-FGF.R
...
Hematop.R.like.complex
Di.hematop.R.complex
GP130.complex
IL2.Rg.complex
KH97.complex
...
Ig.R.like.complex
IL1.R.like.complex
...
G.coupled.R.like.complex
IL8.R.like.complex
...
...
c. Enzyme
P.tyr.kinase
...
P.ser.Thr.kinase
p85 Rsk S6-kinase
p70 S6-kinase
...
...
P.tyr.P.ase
P.ser.Thr.P.ase
...
Phosphorilase
...
Synthetase
...
RNA.Polymerase.II
...
Topoisomerase-II
...
...
Enzyme-complex
...
d. Transcription-factor
Jun.like
c.Jun
Fos
...
NF.kB.like
...
NF.IL2A.like
...
SP1.like
...
TFIID.like
...
TFIIB.like
...
TFIIE.like
...
...

Complex.transcrip-factor
AP.1.like
...
NF.AT.like
...
...
e. Repressor.factor
...
f. Histone
H1
H2A
H2B
...
g. Ribosonal.protein
6S
40S
70S
...
h. Cytoskeleton:protein
Actin
Myosin
Tubulin
Keratin
...
i. Ion.channel
Voltage-gated-channel
Volt-gated-Ca++-channel
LigandGatedChannel
Ca++-gated-K+-channel
...
j. Transplantation.antigen
MHC.I
...
MHC.II
...
k. Adhesion.protein
Integrin
...
Cadherin
...
Selectin
...
...
l. Immunoglobulin
IgG
...
IgM
...
IgA
...
...
m. ECM.protein
Fibronectin
...
Collagen
...
Adhesin
...
...

2. Nucleic.acid
a. DNA
Gene
...
cDNA
...
b. mRNA
...
c. rRNA
...
d. tRNA
...
e. Oligonucleotide
...
3. Carbohydrate
a. Polysacharide
...
4. Lipid
a. Steroid.hormone
Estrogen
...
b. Prostaglandin
...
c. Leukotrienes
...
d. Phospholipid
...
e. Isoprene
...
f. Fatty.acid
...
...
5. Small.molecule
a. Nucleotide
ATP
GTP
cAMP
...
b. Aminoacid
...
c. Vitamin
...
d. Retinoid
...
e. Sugar
...
f. Ion
Cation
...

III. Heter-Mol-Complex
A. TF-DNA.Complex
E2F-RB-DNA.complex
...
B. Nucleosome
...
...

TABLE 168

```
MERGE-BR-LOCATION-FUNCTION-STRUCTURE-PROC (tag1: symbol. type0: symbol. type1:
symbol. type2: symbol. type3: symbol. PETL: class scroll-text-list. QA: class query-array. panel: class
query-panel. win : class window) = (class scroll-text-list. text)

PN: symbol = the name of panel:
BRN: symbol = the related-item of panel:
ETL: class e-scroll-text-list:
 if there exists a e-scroll-text-list ETL such that (the label of ETL =
   "[PN]-[type0]-[type1]-[type2]-[type3]-ETL") then
  go to exit-routine
 else
   create a e-scroll-text-list ETL:
   conclude that the label of ETL = "[PN]-[type0]-[type1]-[type2]-[type3]-ETL":
   conclude that ETL is a-list-associated-to-dlg panel:
   transfer ETL to the subworkspace of safe-home-for-lists at (50. -245):
  TX2 = the scroll-title of ETL:
 if the number of elements in PETL >= 0 then
   if the array-length of QA > 0 then
     TX2 = capitalize-words ("[type2] that are [type0] of [BRN] and located in the [type1] that contain
          [type3]"):
     for T1 = each text in PETL do
      for S2 = each symbol in QA do
       if T1 = "[S2]" then
        insert T1 at the end of the text list ETL:
   else
     if type3 = symbol ("no-[tag1]-selection") then
      TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
         and located in the [type1]")
     else
      TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
         and contained in the [type1] . None of those contain [type3]"):
     for T1 = each text in PETL do
      insert T1 at the end of the text list ETL:
 return ETL. TX2:
exit-routine:
  TX2 = the scroll-title of ETL:
  return ETL. TX2:
```

TABLE 169

```
MERGE-BR-LOCATION-FUNCTION-STRUCTURE-STRUCTURE-PROC (tag1: symbol. tag2:
symbol.
type0: symbol. type1: symbol. type2: symbol. type4: symbol. type5: symbol. PETL: class scroll-text-list.
QA1: class query-array. QA2: class query-array.  panel: class query-panel. win : class window) = (class
scroll-text-list. text)

BRN: symbol = symbol (the related-item of panel):
PN: symbol = the name of panel:
 if there exists a e-scroll-text-list ETL such that (the label of ETL =
   "[PN]-[type0]-[type1]-[type2]-[type4]-[type5]-ETL") then
  go to exit-routine
 else
   create a e-scroll-text-list ETL:
   conclude that the label of ETL =
     "[PN]-[type0]-[type1]-[type2]-[type4]-[type5]-ETL":
   conclude that ETL is a-list-associated-to-dlg panel:
   transfer ETL to the subworkspace of safe-home-for-lists at (50. -245):
   TX2 = the scroll-title of ETL:
```

continue

TABLE 169 cont.

```
  if the array-length of QA1 = 0 then
    if the array-length of QA2 = 0 then
      if type5 = symbol ("no-[tag2]-selection") and type4 = symbol ("no-[tag1]-selection") then
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1]")
      else
       if type5 = symbol ("no-[tag2]-selection") then
        TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of
           [BRN] and located in the [type1]. None contain [type4]")
      else
       if type4 = symbol ("no-[tag1]-selection") then
        TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of
           [BRN] and located in the [type1] contain [type5]")
      else
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1]"):
    for T = each text in PETL do
     insert T at the end of the text list ETL:
   else if the array-length of QA2 >= 0 then
      if type4 = symbol ("no-[tag1]-selection") then
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1] that contain [type5]")
      else
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1] that contain [type5]. None was found also containing [type4]"):
       for T = each text in PETL do
        for S2 = each symbol in QA2 do
         if T = "[S2]" then
          insert T at the end of the text list ETL:
  else if the array-length of QA1 >= 0 then
    if the array-length of QA2 = 0 then
      if type5 = symbol ("no-[tag2]-selection") then
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1] that contain [type4]")
      else
       TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
         and located in the [type1] that contain [type4]. None was found also containing [type5]"):
       for T = each text in PETL do
        for S1 = each symbol in QA1 do
         if T = "[S1]" then
          insert T at the end of the text list ETL:
    else if the array-length of QA2 >= 0 then
      TX2 = capitalize-words ("[type2] with structures defined in this module that are [type0] of [BRN]
          and located in the [type1] that contain [type4] and [type5]"):
       for T = each text in PETL do
        for S1 = each symbol in QA1 do
         if T = "[S1]" then
          for S2 = each symbol in QA2 do
           if T = "[S2]" then
            insert T at the end of the text list ETL:
 return ETL. TX2:
exit-routine:
  TX2 = the scroll-title of ETL:
  return ETL. TX2:
```

TABLE 170

an user-menu-choice

Label	list-copies
Applicable class	complex-bioentity
Condition	the item has a name
Action	start ET-LIST-COPIES-CALLBACK (the item, this window)

```
ET-LIST-LOCAL-COPIES-CALLBACK (ET: class complex-bioentity, win : class window)
  EN: symbol = the name of ET;
  n: integer = 0;
 create a message-dialog md. set the text of the message upon the SW of md to "Processing . . . . .
    and show the SW of md on the current window at (0, 0);
 if there exists a scroll-text-list that is the-copies-list-of ET then
  for CEL = each scroll-text-list that is the-copies-list-of ET do
   delete CEL;
 create a scroll-text-list CEL;
 conclude that CEL is THE-COPIES-LIST-OF ET;
 transfer CEL to the subworkspace of safe-home-for-lists at (55, 15);
 call LIST-LOCAL-BIOENTITY-COPIES-PROC (ET, CEL, win);
 if the number of elements in CEL = 0 then
  go to exit--routine
 else
   create a query-panel panel by cloning MASTER-LOCAL-BIOENTITY-COPIES-PANEL;
   transfer panel to the subworkspace of dialog-bin at (0, 0);
   SW = the subworkspace of panel;
   make panel permanent;
   create a panel-text FT;
   change the text of FT to "COPIES OF: [EN]";
   transfer FT to SW at (0, 80);
   sa = call create-scroll-area-from-list ("ET-copies-scroll", 5, CEL);
   conclude that the allow-unselect-on-selected-message of sa is true;
   conclude that the allow-multiple-simultaneous-selections of sa is true;
   call add-grobj-to-dialog(panel, sa, 0, -30);
   call scroll-area-reflect (sa);
   call set-attribute(SA, the symbol uil-message-selection-method,
      the symbol ET-COPIES-MESSAGE-SELECTION-CALLBACK );
   call uil-set-attribute(SA, the symbol uil-message-unselection-method,
     the symbol ET-COPIES-MESSAGE-UNSELECTION-CALLBACK );
   de = call create-pushbutton ("de", the symbol text, the symbol DELETE-QUERY-PANEL-CALLBACK,
      the symbol medium, "DELETE");
   call add-grobj-to-dialog(panel, de, 0, -145);
   change the size of the subworkspace of panel to minimum;
   show the subworkspace of panel on win with its center 200 units to the left of the center of the screen;
 delete md;
 return;
exit--routine:
 delete md;
 create a message-dialog md. set the text of the message upon the SW of md to "There are no copies of: [EN]"
    and show the SW of md on the current window at (0, 0);
 return;.
```

TABLE 172

```
ET-COPIES-MESSAGE-SELECTION-CALLBACK (MO: class message-object, SA: class scroll-area, win: class
window)
BRN: symbol;
T: text = the text of MO;
 if there exists a bioentity EN such that (the id of EN = T) then
  show EN on win with its center 300 units to the right of the center of
    the screen ;
```

```
ET-COPIES-MESSAGE-UNSELECTION-CALLBACK (MO: class message-object, SA: class scroll-area, win:
class window)
  T: text = the text of MO;
  if there exists a bioentity EN such that (the id of EN = T) then
  hide the workspace of EN on win;
```

TABLE 171

```
LIST-LOCAL-BIOENTITY-COPIES-PROC (ET: class bioentity, CEL:
class scroll-text-list, win: class window)

 EN: symbol = the name of ET:
 CL = the class of ET:
 if ET is a PROTEIN then
    if there exists a protein CE that is of the class named by CL such that
       (the text of the master-bioentity of CE = "[EN]") then
      for CE = each protein that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a PROTEIN-MOTIF then
    if there exists a protein-motif CE that is of the class named by CL such
      that (the text of the master-bioentity of CE = "[EN]" ) then
     for CE = each protein-motif that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a PROTEIN-DOMAIN then
    if there exists a protein-domain CE that is of the class named by CL such
       that (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each protein-domain that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a PROTEIN-COMPONENT then
    if there exists a protein-component CE that is of the class named by CL
         such that (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each protein-component that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a DNA-COMPONENT then
    if there exists a dna-component CE that is of the class named by CL such
       that (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each dna-component that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a HETER.MOL.COMPLEX then
    if there exists a heter.mol.complex CE that is of the class named by CL
         such that (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each heter.mol.complex that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a NUCLEIC-ACID then
    if there exists a nucleic-acid CE that is of the class named by CL such
       that (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each nucleic-acid that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a MEMBRANE then
    if there exists a membrane CE that is of the class named by CL such that
       (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each membrane that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL: else
 if ET is a BIOENTITY then
    if there exists a bioentity CE that is of the class named by CL such that
       (the text of the master-bioentity of CE = "[EN]" ) then
      for CE = each bioentity that is of the class named by CL do
       if the text of the master-bioentity of CE = "[EN]" then
        insert "[the id of CE]" at the end of the text list CEL:
 return:
```

TABLE 173

an user-menu-choice

Label	show-lists
Applicable class	bioreservoir
Condition	the br-chaining-tracer upon the subworkspace of the item exits
Action	start BR-CHAINING-PROC (the item. this window)

```
BR-CHAINING-PROC (BR: class bioreservoir. win: class window)
 tracer: = the br-chaining-tracer upon the subworkspace of BR:
 if the subworkspace of tracer exists then
   show the subworkspace of tracer on Win with its center 400 units to the right of
      the center of the screen
 else
   if BR has a name then go to chaining-routine
   else if the master-bioreservoir of BR is not none then
     BR = the bioreservoir named by the master-bioreservoir of BR:
     go to chaining-routine:
   else return:
chaining-routine:
   create a workspace SW by cloning the subworkspace of BR-CHAINING-LISTS :
   make SW the subworkspace of tracer:
   change the text of the panel-text upon SW to "Lists of bioReservoirs and bioProcesses
      Downstream and Upstream of: [the name of BR]":
   DBPL = the downstream-bioprocesses-list upon SW :
   DBRL = the downstream-bioreservoirs-list upon SW :
   for DBP = each bioprocess that is a-down-bioprocess-of BR do
     allow other processing:
     insert DBP at the end of the bioprocess list DBPL:
     for DDBP = each bioprocess that is a-downstream-bioprocess-of DBP do
       insert DDBP at the end of the bioprocess list DBPL:
    for LDDBP = each bioprocess in DBPL do
      for DDBR = each bioreservoir that is a-down-bioreservoir-of LDDBP do
        insert DDBR at the end of the bioreservoir list DBRL:
   UBPL = the upstream-bioprocesses-list upon SW :
   UBRL = the upstream-bioreservoirs-list upon SW :
   for UBP = each bioprocess that is an-up-bioprocess-of BR do
     allow other processing:
     insert UBP at the end of the bioprocess list UBPL:
     for UUBP=each bioprocess that is an-upstream-bioprocess-of UBP do
       insert UUBP at the end of the bioprocess list UBPL:
    for LUUBP = each bioprocess in UBPL do
     for UUBR =each bioreservoir that is an-up-bioreservoir-of LUUBP do
       insert UUBR at the end of the bioreservoir list UBRL:
   change the size of SW to minimum:
   show SW on win with its center 400 units to the right of the center of the screen:
 return:
```

TABLE 174

an user-menu-choice

Label	show-lists
Applicable class	bioprocess
Condition	the bp-chaining-tracer upon the subworkspace of the item exits
Action	start BP-CHAINING-PROC (the item. this window)

```
BP-CHAINING-PROC (BP: class bioprocess. win: class window )
 tracer= the bp-chaining-tracer upon the subworkspace of BP:
 if the subworkspace of tracer exists then
  show the subworkspace of tracer on win with its center 400 units to the right of the
    center of the screen
 else
  if BP has a name then go to chaining-routine
  else if the master-bioprocessof BP is not none then
    BP = the bioprocess named by the master-bioprocess of BP:
    go to chaining-routine:
  else return:
chaining-routine:
    create a workspace SW by cloning the subworkspace of BP-CHAINING-LISTS:
    make SW the subworkspace of tracer:
    change the text of the panel-text upon SW to "[the name of BP]":
    DBPL = the downstream-bioprocesses-list upon SW :
    SDBPL = the downstream-bp-simulation-list upon SW :
    DBRL = the downstream-bioreservoirs-list upon SW :
    SDBRL = the downstream-br-simulation-list upon SW :
    for DBP = each bioprocess that is a-downstream-bioprocess-of BP do
      allow other processing:
      insert DBP at the end of the bioprocess list DBPL:
      if the status of DBP is downstream-activated or the status of DBP is forw-activated then
          insert DBP at the end of the bioprocess list SDBPL:
      for DBR = each bioreservoir that is a-down-bioreservoir- of DBP do
          insert DBR at the end of the bioreservoir list DBRL:
    UBPL = the upstream-bioprocesses-list upon SW :
    SUBPL = the upstream-bp-simulation-list upon SW :
    UBRL = the upstream-bioreservoirs-list upon SW :
    SUBRL = the upstream-br-simulation-list upon SW :
    for UBP = each bioprocess that is an-upstream-bioprocess-of BP do
      allow other processing:
      insert UBP at the end of the bioprocess list UBPL:
      if the status of UBP is upstream-activated or the status of UBP is back-activated then
        insert UBP at the end of the bioprocess list SUBPL:
      for UBR = each bioreservoir that is an-up-bioreservoir-of UBP do
        insert UBR at the end of the bioreservoir list UBRL:
     change the size of SW to minimum:
     show SW on win with its center 400 units to the right of the center of the screen:
  return:
```

TABLE 175

	an user-menu-choice
Label	up-pathway
Applicable class	bioprocess
Condition	the subworkspace of the item has been activated and the subworkspace of the up-path-tracer upon the subworkspace of the item does not exist
Action	start DRAW-UP-PATHWAY-PROC (the up-path-tracer upon the subworkspace of the item, this window)

```
DRAW-UP-PATHWAY-PROC (PT: class up-path-tracer, win: class window)
 BP = the bioprocess superior to the workspace of PT;
 if BP has a name and there exists a bioreservoir that is an-up-bioreservoir-of BP then
   create a workspace SW by cloning the subworkspace of MASTER-UP-PATHWAY;
   make SW the subworkspace of PT;
   T = the biomodel-title upon SW;
   conclude that the label of T = "[the label of T] [the label of BP]";
   change the status-color icon-color of BP to brown;
   create a bioprocess LBP by cloning BP;
   make the subworkspace of LBP transient;
   delete the subworkspace of LBP;
   x = the x-pos upon SW;
   y = the y-pos upon SW;
   conclude that x = -305;
   conclude that y = -240;
   transfer LBP to SW at (x, y);
   show SW at three-quarter scale with its bottom left corner 50 units to the right of the
     bottom left corner of the screen;
   for UBP = each bioprocess that is an-upstream-bioprocess-of BP do
     allow other processing;
     change the status-color icon-color of UBP to brown;
   conclude that y = y + 30;
   call CREATE-LOCAL-UP-BP-PROC (BP, LBP, SW, win);
  change the size of SW to minimum;
  return.
```

TABLE 176

```
CREATE-LOCAL-UP-BP-PROC (BP: class bioprocess, LBP: class bioprocess, SW: class workspace, win: class window)
 x = the x-pos upon SW;  y = the y-pos upon SW;
 if there exists a bioreservoir that is an-up-bioreservoir-of BP then
   for UBR = each bioreservoir that is an-up-bioreservoir-of BP do
    if there exists a bioprocess that is an-up-bioprocess-of UBR then
      for UBP = each bioprocess that is an-up-bioprocess-of UBR do
       if there exists a bioprocess MBP upon SW such that (the master-bioprocess of MBP = the master-bioprocess
           of UBP) then LUBP = MBP;
         go to connect-routine;
       else
        conclude that x = x + 110;
        if x > 2600 then show SW at one-quarter scale with its top left corner 50 units below the top left corner of the screen
 else
        if x > 1800 then show SW at half scale with its top left corner 50 units below the top left corner of the screen.
        create a bioprocess LUBP by cloning UBP;
        make the subworkspace of LUBP transient;
        delete the subworkspace of LUBP;
        transfer LUBP to SW at (x, y);
        r = random (20, 76);   l = random (5, 25);
        create a connection connected between LBP newly locating it at top at the position given by r and LUBP newly l
          ocating it at left at the position given by l with direction input;
        conclude that y = y + 30;
        call CREATE-LOCAL-UP-BP-PROC (UBP, LUBP, SW, win);
 return;
 connect-routine:
   for C = each connection connected to LBP do
    if there exists a connection UC connected to LUBP such that (UC is the same object as C) then return;
   b = random (20, 76);
   r = random (40, 51);
   p = x-rel-position (LBP, LUBP);
   if p > 0 then
    create a connection of class e-line connected between LBP newly locating it at bottom at the position given by b
       and LUBP newly locating it at right at the position given by r with direction input , with vertices 10 else
   if p < 0 then
    create a connection of class e-line connected between LBP newly locating it at right at the position given by r
       and LUBP newly locating it at bottom at the position given by b with direction input ;
```

TABLE 177

	an user-menu-choice
Label	navig-panel
Applicable class	bioreservoir
Condition	the item has a name and the subworkspace of the item exists
Action	start CREATE-DIRECTION-PANEL-CALLBACK (the item, this window)

	an user-menu-choice
Label	navig-panel
Applicable class	biopool
Condition	the bioreservoir superior to the workspace of the item has a name
Action	start CREATE-DIRECTION-PANEL-CALLBACK (the bioreservoir of the biopost connected to the item, this window)

	an user-menu-choice
Label	navig-panel
Applicable class	biorole-object
Condition	none
Action	start CREATE-DIRECTION-PANEL-CALLBACK (the ref-bioreservoir superior to the workspace of the item, this window)

```
CREATE-DIRECTION-PANEL-CALLBACK (BR: class bioreservoir, win: class window)
 BRN: symbol = the name of BR:
 create a panel by cloning MASTER-NAVIG-DIRECTION-PANEL:
 transfer panel to the subworkspace of DIALOG-BIN
 SW = the subworkspace of panel:
 call SET-RADIO-BOX-ID-PROC ("navig-direction", "[BRN]-NAVIG-DIRECTION-radio-box", SW, win):
 create a panel-text FT:
 change the text of FT to "DIRECTION OF NAVIGATION FROM: [BRN]":
 transfer FT to SW at (0, 110):
 call shrink-wrap-dialog(panel):
 change the text of the related-item of panel to "[BRN]":
 show SW on win at (0, 0):
 return:
```

TABLE 178

```
SELECT-DIRECTION-CALLBACK (button: class button, win : class window, dlg: class dialog)
 BR = the bioreservoir named by the related-item of dlg:
 rb-id = call GET-RADIO-BOX-ID-PROC ("navig-direction", the subworkspace of dlg, win):
 type = call get-radio-box-value (dlg, rb-id):
 call delete-dialog (dlg):
 md = call create-message-dialog ("md", "Processing  . . . .  ", the symbol extra-large, win, 0, 0):
 case (type) of
   "Upstream": panel = call CREATE-UPSTREAM-NAVIG-PANEL-CALLBACK (BR, win):
   "Downstream": panel = call CREATE-DOWNSTREAM-NAVIG-PANEL-CALLBACK (BR, win):
 call delete-dialog (dlg):
 delete md:
```

TABLE 179

```
CREATE-DOWNSTREAM-NAVIG-PANEL-CALLBACK (BR: class bioreservoir ,win : class window)
   = (class entry-panel)
 BRN: symbol = the name of BR:
 if not (there exists a bioprocess that is a-down-bioprocess-of BR) then go to exit-routine:
 panel = call clone-panel (MASTER-DOWN-NAVIGATION-PANEL):
 change the text of the related-item of panel to "[BRN]":
 create a panel-text FT:
 change the text of FT to "DOWNSTREAM NAVIGATION PANEL FOR: [BRN]":
 transfer FT to the subworkspace of panel at (70, 360):
 call NAVIG-CREATE-DOWNSTREAM-LISTS-PROC (BR, panel, win):
 DBPL= the bioprocesses-list that is the-dsbp-list-of panel:
 make panel permanent:
 create a text-list PSML:
 for DBP = each bioprocess that is a-down-bioprocess-of BR do
   for DBM = each biomodel that is the-container-of-bp DBP do
     if DBM is not a bioreservoir-bin then insert "[the name of DBM]" at the end of the text list PSML:
 call COMPLETE-NAVIG-PANEL-CALLBACK (BR, PSML, panel,win):
 return panel:
exit-routine:
   md = call create-message-dialog ("md", "There is no down-bioprocess of [the name of BR]".
     the symbol extra-large, win, 0, 0):
   return md:
```

TABLE 180

```
NAVIG-CREATE-DOWNSTREAM-LISTS-PROC (BR: class bioreservoir.
panel: class entry-panel. win: class window)
 if there exists a downstream-bioprocesses-list that is the-dsbp-list-of panel then
  for BPL = each bioprocesses-list that is the-dsbp-list-of panel do
   delete BPL:
 if there exists a downstream-bioreservoirs-list that is the-dsbr-list-of panel then
  for BRL = each downstream-bioreservoirs-list that is the-dsbr-list-of panel do
   delete BRL:
 allow other processing:
 create a downstream-bioreservoirs-list BRL:
 conclude that BRL is THE-DSBR-LIST-OF panel :
 transfer BRL to the subworkspace of safe-home-for-lists:
 create a downstream-bioprocesses-list BPL:
 conclude that BPL is THE-DSBP-LIST-OF panel :
 transfer BPL to the subworkspace of safe-home-for-lists:
 insert BR at the end of the bioreservoir list BRL:
 if there exists a bioreservoir DBR such that (DBR is a-downstream-bioreservoir-of BR) then
  for DBR = each bioreservoir that is a-downstream-bioreservoir-of BR do
   insert DBR at the end of the bioreservoir list BRL:
 for DBP = each bioprocess that is a-down-bioprocess-of BR do
  insert DBP at the end of the bioprocess list BPL:
  if there exists a bioprocess DSBP such that (DSBP is a-downstream-bioprocess-of DBP) then
   for DSBP = each bioprocess that is a-downstream-bioprocess-of DBP do
    insert DSBP at the end of the bioprocess list BPL:
 return:
```

TABLE 181

```
LIST-NAVIGATION-DBP-CALLBACK (button: class button. win: class window.
panel: class entry-panel)
   SW = the subworkspace of panel :
   BRN= the related-item of panel:
   BR= the bioreservoir named by BRN:
   BRL= the downstream-bioreservoirs-list that is the-dsbr-list-of panel :
   BPL = the downstream-bioprocesses-list that is the-dsbp-list-of panel :
   SA = the scroll-area upon SW:
  if not (there exists an uil-message-object that is a-selected-message-ofSA) then
   go to no-entry-routine:
  create a bm-downstream-bp-simulation-list DBPL:
  create a down-br-simulation-list DBRL:
  call NAVIG-READ-DOWN-SCROLL-PROC (BPL. BRL. DBPL. DBRL. win. panel ):
  n = the number of elements in DBPL:
  conclude that the lenght of DBPL = n:
  transfer DBPL to SW at (100. 20):
  transfer DBRL to SW at (170. 20):
  de = call get-obj-from-id ("de". panel):
  call disable-button (de):
  call disable-button (button):
  return:
 no-entry-routine:
    md = call create-message-dialog ("md". "No bioModel has been selected for this navigation.
      Click on the ? button. follow the instructions and click on LIST again". the symbol large).
    win, 0. 0):
    return:
```

TABLE 182

```
NAVIG-READ-DOWN-SCROLL-PROC (BPL: class downstream-bioprocesses-list. BRL:
class downstream-bioreservoirs-list. DBPL: class bm-downstream-bp-simulation-list.
DBRL: class down-br-simulation-list. win: class window. panel: class entry-panel)
 SW = the subworkspace of panel:
 scroll-array =call get-selected-message (panel. "select-biomodel-scroll"):
 for ST = each text in scroll-array do
  BMN = symbol (ST):
  BM = the biomodel named by BMN:
  conclude that BM is THE-NAVIGATED-BM-OF panel:
  n = n + 1:
  if n = 1 then
   SBMT = ST else
  if n > 1 then
   SBMT = "[SBMT]. [ST]":
  show the subworkspace of BM at three-quarter scale with its top right corner 40 units below
      the top right corner of the screen:
 for DBR = each bioreservoir in BRL do
  allow other processing:
  if DBR is BR-CONTAINED-IN BM then insert DBR at the end of the bioreservoir list DBRL:
 for DBP = each bioprocess in BPL do
  allow other processing:
  if DBP is BP-CONTAINED-IN BM then insert DBP at the end of the bioprocess list DBPL:
 change the name of DBPL to symbol("downstream-BP-within-[SBMT]"):
 change the name of DBRL to symbol("downstream-BR-within[SBMT]"):
 conclude that the ref-biomodel of DBPL = "[SBMT]":
 dr = call get-obj-from-id ("dr". panel):
 call enable-button (dr):.
 return:
```

TABLE 183

```
NAVIG-DRAW-DBP-CALLBACK (button: class button. win: class item. panel: class entry-panel)
 WS: class kb-workspace = the subworkspace of panel:
 DBPL = the bm-downstream-bp-simulation-list upon WS:
 SW = call CREATE-NAVIG-PATH-TRACER-PROC ("downstream". panel. win):
 show SW at three-quarter scale with its top left corner 50 units below the top left corner of the screen:
 BR = the bioreservoir named by the related-item of panel:
 if there exists a bioprocess that is a-down-bioprocess-of BR then
     BT = the biomodel-title upon SW:
     conclude that the label of BT = "[the label of BT]: [the label of BR]":
     change the status-color icon-color of BR to brown:
     x = the x-pos upon SW:
     y = the y-pos upon SW:
     conclude that x = - 50:
     conclude that y = 120:
     create a bioreservoir LBR by cloning BR:
     delete the subworkspace of LBR:
     transfer LBR to SW at (x. y):
     conclude that y = y - 40:
     for DBP = each bioprocess that is a-down-bioprocess-of BR do
      if DBP is a member of the bioprocess list DBPL then
         allow other processing:
         conclude that x = x + 110:
         create a bioprocess LDBP by cloning DBP:
         delete the subworkspace of LDBP:
         transfer LDBP to SW at (x. y):
         r = random (3. 35):
         t = random (20. 76):
         create a connection connected between LBR newly locating it at right at the position given by r
             and LDBP newly locating it at top at the position given by t with direction output:
         conclude that y = y - 40:
         call NAVIG-CREATE-LOCAL-DBP-PROC (DBP. LDBP. DBPL. SW. win):
 change the size of SW to minimum:
 call disable-button (button):.
 return:
```

TABLE 184

```
NAVIG-CREATE-LOCAL-DBP-PROC (BP: class bioprocess. LBP: class bioprocess. DBPL:
class bm-downstream-bp-simulation-list. SW: class workspace. win: class window)

 x = the x-pos upon SW:
 y = the y-pos upon SW:
 if there exists a bioreservoir that is a-down-bioreservoir-of BP then
  for BR = each bioreservoir that is a-down-bioreservoir-of BP do
    if there exists a bioprocess that is a-down-bioprocess-of BR then
    for DBP = each bioprocess that is a-down-bioprocess-of BR do
     if DBP is a member of the bioprocess list DBPL then
        if there exists a bioprocess MBP upon SW such that (the master-
            bioprocess of MBP = the master-bioprocess of DBP) then
          LDBP = MBP:
          call CONNECT-EXISTING-LOCAL-DBP-PROC (LBP, LDBP, win):
        else
          allow other processing:
          conclude that x = x + 110:
          if x > 2600 then show SW at one-quarter scale with its top
            left corner 50 units below the top left corner of the
            screen else
          if x > 1800 then show SW at half scale with its top left
            corner 50 units below the top left corner of the
            screen:
          create a bioprocess LDBP by cloning DBP:
          delete the subworkspace of LDBP:
          transfer LDBP to SW at (x. y):
          r = random (3. 35):
          t = random (20. 76):
          create a connection connected between LBP newly locating
              it at right at the position given by r and LDBP newly
              locating it at top at the position given by t with
              direction output:
          conclude that y = y - 40:
          call NAVIG-CREATE-LOCAL-DBP-PROC (DBP, LDBP. DBPL. SW. win):
 return:
```

```
CONNECT-EXISTING-LOCAL-DBP-PROC (LBP: class bioprocess. LDBP: class ioprocess.
win: class window)

  for C = each connection connected to LBP do
    if there exists a connection DC connected to LDBP such that (DC is the
      same object as C) then return:
   b = random (20. 76):
   r = random (3, 35):
   p = x-rel-position (LBP. LDBP):
   if p > 0 then
    create a connection of class c-line connected between LBP newly locating it at right at the
position given by r and LDBP newly locating it at bottom at the position given by b with direction
output. with vertices 6 -60 else
   if p < 0 then
    create a connection of class c-line connected between LBP newly
       locating it at bottom at the position given by b and LDBP newly
       locating it at left 52 with direction output. with vertices -60:
   return:
```

TABLE 185

```
CREATE-NAVIG-PATH-TRACER-PROC (type: text, panel: class entry-panel, win: class item) = (class workspace)
 WS: class kb-workspace = the subworkspace of panel:
 if there exists a navig-path-tracer upon WS then
    for NT = each navig-path-tracer upon WS do
       delete NT:
 if type = "downstream" then
   create a navig-path-tracer NT by cloning MASTER-DOWN-NAVIG-PATHWAY else
 if type = "upstream" then
  create a navig-path-tracer NT by cloning MASTER-UP-NAVIG-PATHWAY:
 transfer NT to WS at (130, - 85):
 SW = the subworkspace of NT:
 show WS on win with its bottom right corner at the bottom right corner of the screen:
 call configure-path-tracer-proc (NT, win):
 return SW:
```

TABLE 186

```
COMPLETE-NAVIG-PANEL-CALLBACK (BR: class bioreservoi, PSML: class text-list, panel: class entry-panel,
win : class window)
 SW = the subworkspace of panel:
 sa = call create-scroll-area-from-list ("select-biomodel-scroll", 5, PSML, panel, 70, 170):
 conclude that the allow-unselect-on-selected-message of sa is true:
 conclude that the message-selection-method of sa is the symbol MESSAGE-SELECTION:
 conclude that the message-unselection-method of sa is the symbol MESSAGE-UNSELECTION:
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
   change the text of the callback of de to "DELETE-ENTRY-PANEL-CALLBACK":
 change the size of panel to minimum
 show SW on win at the center of the screen:
```

TABLE 187

```
RESET-NAVIGATION-CALLBACK (button: class button, win: class item, panel: class entry-panel)
 call RESET-NAVIG-PROC (panel):
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then enable de:
```

```
RESET-NAVIG-PROC (panel: class e-entry-panel)
SW: class kb-workspace = the subworkspace of panel:
 if there exists a biomodel that is THE-NAVIGATED-BM-OF panel then
  for BM = each biomodel that is THE-NAVIGATED-BM-OF panel do
    allow other processing:
    change the status-color icon-color of BM to lime-green:
    conclude that BM is not THE-NAVIGATED-BM-OF panel :
 if there exists a navig-path-tracer upon SW then
  for NT = each navig-path-tracer upon SW do
    delete NT:
 if there exists a bioprocesses-list BPL upon SW then
    for BP = each bioprocess in BPL do
       allow other processing:
       conclude that the status of BP is available:
       change the status-color icon-color of BP to slate-blue:
    delete BPL:
 if there exists a bioreservoirs-list BRL upon SW then
    for BR = each bioreservoir in BRL do
       allow other processing:
       conclude that the status of BR is available:
       change the status-color icon-color of BR to medium-aquamarine:
    delete BRL:
    if there exists a text-pushbutton ls upon SW such that (the id of ls = "ls ") then enable ls.
    if there exists a text-pushbutton dr upon SW such that (the id of dr = "dr ") then disable dr:
```

TABLE 188

```
SIMUL-INIT-CALLBACK (button: class button. win: class window. panel: class selection-panel)
 call GENERAL-INIT-PROC (win):
 call SIMUL-INIT-PROC (win):
 return:
---
ADD-SIMUL-CALLBACK (button: class uil-button. win: class window. panel: class selection-panel)
 call SIMUL-INIT-PROC (win):
return:
---
SIMUL-INIT-PROC (win: class g2-window)
SW = the subworkspace of INITIALIZATION-PANEL:
 call SEND-NO-INPUT-MESSAGES (win):
 if there exists a sd-br-list upon SW then
  for NL = each sd-br-list upon SW do
   delete NL:
 create a sd-br-list UNL: conclude that the label of UNL = "UNList":
 create a sd-br-list ANL: conclude that the label of ANL = "ANList":
 create a sd-br-list RNL: conclude that the label of RNL = "RNList":
 create a sd-br-list DNL: conclude that the label of DNL = "DNList":
 for BR = each bioreservoir do
  if BR has a name then
   SD = the scaling-density of the biopool upon the subworkspace of BR:
   if the set-by-user of SD is true then insert BR at the end of the bioreservoir list UNL
   else call SET-SCALING-DENSITY-PROC (BR. ANL. RNL. DNL)
 transfer UNL to SW at (- 130. -235):
 transfer ANL to SW at (- 80. -235):
 transfer RNL to SW at (60. -235):
 transfer DNL to SW at (110. -235):
 inform the operator that "The setting of the scaling-density of all bioPools has been concluded":
 change the size of SW to minimum:
 return:
---
SEND-NO-INPUT-MESSAGES (win: class item)
 for BR = each bioreservoir do
  exit if BR has no name:
  PO = the biopool upon the subworkspace of BR:
  if not (there exists a bioproduct connected to PO) and not (there exists a model-block M connected to PO) then
   create a message-dialog md. change the text of the message upon theSW of md to "The bioPool of [the name of BR]
    has no input !!! To model an input source: 1. Select 'create sw' from the menu of the scaled-input-model-boxor the
    input-model-box of this bioPool. as desired. 2. Add / connect / configure an appropriate model-block" and show
    the SW of md on the current window at (0. 0).
   change the text of the callback of the text-pushbutton upon the subworkspace of md to "DELETE-DIALOG-CALLBACK":
---
SET-SCALING-DENSITY-PROC (BR: class bioreservoir. ANL: class sd-br-list. RNL: class sd-br-list.
   DNL: class sd-br-list)
PO = the biopool upon the subworkspace of BR:
SDF = the scaling-density of PO:
 if the if-scaling-amount of BR /=100 then
   insert BR at the end of the bioreservoir list ANL:
   sa = the if-scaling-amount of BR:
   change the text of the initial-value of the scaling-density of PO to "[sa]":
   make the scaling-density of PO permanent:
   if the if-scaling-bioreservoir of BR is not none then go to conflict-routine:
 else
 if the if-scaling-bioreservoir of BR is not none then
   insert BR at the end of the bioreservoir list RNL:
   sr = the if-scaling-amount of the bioreservoir named by the if-scaling-bioreservoir of BR:
   change the text of the initial-value of the scaling-density of PO to "[sr]":
   make the scaling-density of PO permanent:
 else
 if the if-scaling-amount of BR =100 then
   insert BR at the end of the bioreservoir list DNL:
   change the text of the initial-value of the scaling-density of PO to "100.0":
   make the scaling-density of PO permanent:
 return:
conflict-routine:
 sr = the if-scaling-amount of the bioreservoir named by the if-scaling-bioreservoir of BR:
 if sa /= sr then inform the operator that "The attributes if-scaling-amount and if-scaling-bioreservoir of [the name of BR]
   have both values. which provide two conflicting scaling-density values for its bioPool: [sa] and [sr]. respectively.
   The initial value of scaling-density has been set to [sa]. but it is recommended that the conflict is resolved. and this
   value changed if desired.".
  return:
---
```

alternative method for mixed absolute/scaled reasoning

TABLE 189

```
long-SET-SCALING-DENSITY-PROC (BR: class bioreservoir, NSDL:
class bioreservoirs-list)

if the set-by-user of the scaling-density of the PO of BR is true then
  return else do:
  create a float-list LeL;
  define an internal float value SC;
  define an internal float value SD;
  for each bioproduct P connected to PO do
    create a float-list PL;
    for each symbol PREC in the scaling-precursors of P do:
      if the biorole-object R named by PREC is not an amplifier-
          bioreactant or an inhibitor.bioreactant and if the scaling-density
          of the biopool UPO connected to R /= 99.999 then
        insert the scaling-density of UPO at the end of PL;
    if the number of elements in PL >= 1 then
      insert the minimum float value in PL at the end of the float list LeL ;
    delete PL;
    if the number of elements in LeL >= 1 then:
      set SD equal to the maximum float value in LeL;
      go to set-routine;
    else if there exists a source-bioreactant connected to PO then:
      create a float-list TSL;
      for each source-bioreactant TS connected to PO do
        if TS is a SUBSTRATE.R and the MICHAELIS-CONSTANT of
            TS /= 9.9e-99 then
          insert the MICHAELIS-CONSTANT of TS at the end of TSL
        else if TS is an ION.R and the EQUILIBRIUM-CONSTANT of
            TS /= 9.9e-99 then
          insert the EQUILIBRIUM-CONSTANT of TS at the end of
          TSL;
      if the number of elements in TSL >= 1 then
        set SC equal to the average of all floats in TSL;
    else if there exists a binding-bioreactant R connected to PO such that
        the EQUILIBRIUM-DISSOCIATION-CONSTANT of
        R /= 9.9e-99 then:
      create a float-list TSL;
      for each binding-bioreactant Ts connected to PO do
        if the EQUILIBRIUM-DISSOCIATION-CONSTANT of
            Ts /= 9.9e-99 then
          insert the EQUILIBRIUM-DISSOCIATION-CONSTANT of Ts
            at the end of TSL;
      if the number of elements in TSL >= 1 then
        set SC equal to the average of all floats in TSL;
    if SC /= 0.0 then:
      SD = SC * 6.023e23 * top-volume-br (BR);
      go to set-routine;
  if LeL exists then delete LeL;
  return

set-routine:
  if SD = 0.0 then
    insert BR at the end of the bioreservoir list NSDL
  else if SD /= 0.0 then
    if the scaling-density of PO = 99.999 then
      set the initial-value of the scaling-density of PO = the value of SD
    else if the scaling-density of PO /= 99.999 and the scaling-density of
      PO /= SD then inform the user.
  return
```

TABLE 190

an user-menu-choice
Label simul-panel
Applicable class bioreservoir
Condition the item has a name and the subworkspace of the item exists
Action start CREATE-INPUT-TYPE-CALLBACK (the item, this window)

---

an user-menu-choice
Label simul-panel
Applicable class biopool
Condition the bioreservoir superior to the workspace of the item has a name
Action start CREATE-INPUT-TYPE-CALLBACK
(the bioreservoir superior to the workspace of the item, this window)

---

an user-menu-choice
Label simul-panel
Applicable class biorole-object
Condition the biopost connected to the item has a name
Action start CREATE-INPUT-TYPE-CALLBACK (the bioreservoir named by the
ref-bioreservoir of the biopost connected to the item, this window)

```
CREATE-INPUT-TYPE-CALLBACK (BR: class bioreservoir, win: class window)
 WS = the subworkspace of BR;
 BRN: symbol = the name of BR;
 if not (there exists a bioprocess that is a-down-bioprocess-of BR) then
  go to exit-routine;
 activate the subworkspace of BR;
 PO = the biopool upon WS;
 call CREATE-DOWNSTREAM-LISTS-PROC (BR, win);
 if PO is a sol-mol-pool or PO is a exp-cell-pool then
    panel = call clone-panel (SIMUL-CONCENTRATION-INPUT-TYPE-PANEL);
    SW = the subworkspace of panel;
    call SET-RADIO-BOX-ID-PROC ("simul-conc", "[BRN]-INPUT-TYPE-radio-box", SW, win);
 else
    panel = call clone-panel (SIMUL-DENSITY-INPUT-TYPE-PANEL);
    SW = the subworkspace of panel;
    call SET-RADIO-BOX-ID-PROC ("simul-density", "[BRN]-INPUT-
       TYPE-radio-box", SW, win);
 create a panel-text FT;
 change the text of FT to "INPUT TYPE FOR: [BRN]".
 transfer FT to SW at (0, 110);
 change the size of SW to minimum
 change the text of the related-item of panel to "[BRN]";
 show SW on win at the center of the screen;
 return;
exit-routine:
   md = call create-message-dialog ("md", "There is no down-bioprocess-of
       [the name of BR]", the symbol extra-large, win, 0, 0);
   return;
```

```
CREATE-DOWNSTREAM-LISTS-PROC (BR: class bioreservoir, win: class window)
 if there exists a downstream-bioprocesses-list that is the-downstream-
    bioprocesses-list-of BR then
   for DBPL = each bioprocesses-list that is the-downstream-bioprocesses-
       list-of BR do
    delete DBPL;
 if there exists a downstream-bioreservoirs-list that is the-downstream-
    bioreservoirs-list-of BR then
   for DBRL = each downstream-bioreservoirs-list that is the-downstream-
       bioreservoirs-list-of BR do
    delete BRL;
 if there exists an up-bioreservoirs-list that is the-up-bioreservoirs-list-of BR then
   for UBRL = each downstream-bioreservoirs-list that is the-downstream-bioreservoirs-list-of BR do
    delete UBRL;
 allow other processing;
 create a downstream-bioreservoirs-list DBRL;
 conclude that DBRL is the-downstream-bioreservoirs-list-of BR;
 transfer DBRL to the subworkspace of safe-home-for-lists;
 create a downstream-bioprocesses-list DBPL;
 conclude that DBPL is the-downstream-bioprocesses-list-of BR;
 transfer DBPL to the subworkspace of safe-home-for-lists;
 create an-up-bioreservoirs-list UBRL;
 conclude that UBRL is the-up-bioreservoirs-list-of BR;
 transfer UBRL to the subworkspace of safe-home-for-lists;
 call FILL-BR-DOWNSTREAM-LISTS-PROC (BR, DBRL, DBPL, DBRL, win);
 return;
```

TABLE 191

```
SELECT-INPUT-PANEL-CALLBACK (button: class button. win : class window. dlg: class dialog)
 BRN = the related-item of dlg:
 BR = the bioreservoir named by BRN:
 WS = the subworkspace of dlg:
 rb-id = call GET-RADIO-BOX-ID-PROC ("simul-selection". WS. win):
 type = call get-radio-box-value (dlg. rb-id):
 call delete-dialog (dlg):
 md = call create-message-dialog ("md". "Processing . . . . .  ". the symbol extra-large. win. 0. 0):
 case (type) of
  "one-time-concentration":
   panel = call CREATE-CONC-INPUT-PANEL-PROC (BR. win):
  "one-time-density":
   panel = call CREATE-DENSITY-INPUT-PANEL-PROC (BR. win):
  "one-time-relative":
   panel = call CREATE-RELATIVE-INPUT-PANEL-PROC (BR. win):
  "periodic-concentration":
   panel = call CREATE-PERIODIC-CONC-INPUT-PANEL-PROC (BR. win):
  "periodic-density":
   panel = call CREATE-PERIODIC-DENSITY-INPUT-PANEL-PROC (BR. win):
  "periodic-relative":
   panel = call CREATE-PERIODIC-RELATIVE-INPUT-PANEL-PROC (BR. win):
 make panel permanent:
 change the text of the related-item of panel to "[BRN]":
 SW = the subworkspace of panel:
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
   change the text of the callback of de to "DELETE-ENTRY-PANEL-CALLBACK":
 change the size of SW to minimum
 show SW on win at the center of the screen:
 return panel:
 delete md:
```

TABLE 192

```
COMPLETE-INPUT-PANEL-PROC (BR: class bioreservoir. panel: class entry-panel. win : class window) =
(class entry-panel)
 SW = the subworkspace of panel:
 BRN = the name of BR:
 create a text-list PSML:
 for DBP = each bioprocess that is a-down-bioprocess-of BR do
  for DBM = each biomodel that is the-container-of-bp DBP do
   if DBM is not a bioreservoir-bin then
    insert "[the name of DBM]" at the end of the text list PSML:
 sa = call create-scroll-area-from-list ("select-biomodel-scroll". 5. PSML. panel. 400. 180):
 conclude that the allow-unselect-on-selected-message of sa is true:
 conclude that the allow-multiple-simultaneous-selections of sa is true:
 conclude that the allow-multiple-simultaneous-selections of sa is true:
 change the text of the uil-message-selection-method to "message-selection":
 change the text of the uil-message-unselection-method to "message-unselection":
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then
   change the text of the callback of de to "DELETE-ENTRY-PANEL-CALLBACK":
 change the size of SW to minimum:
 show SW on win at the center of the screen:
 return panel:
```

TABLE 193

```
CREATE-PERIODIC-RELATIVE-INPUT-PANEL-PROC (BR: class bioreservoir. win: class window) = (class entry-panel)
 BRN = the name of BR:
 panel = call clone-panel (MASTER-PERIODIC-RELATIVE-ENTRY-PANEL):
 SW = the subworkspace of panel:
 create a panel-text FT and transfer FT to SW at (-20. 380):
 change the text of the id of panel to "[BRN]-periodic-relative-entry-panel":
 change the text of FT to "1 PERIODIC RELATIVE INPUT FOR: [BRN]":
 ac = call create-pushbutton ("ac". the symbol ACTIVATE-RELATIVE-INPUT-CALLBACK.
    "ACTIVATE". panel. 490. 65):
 disable ac:
 st = call create-pushbutton ("st". the symbol START-PERIODIC-RELATIVE-INPUT-CALLBACK.
    "START". panel. 140. 65):
 disable st:
 return panel:
```

TABLE 194

```
FILL-BR-DOWNSTREAM-LISTS-PROC (BR: class bioreservoir, DBRL: class downstream-bioreservoirs-list, DBPL: class
downstream-bioprocesses-list, UBRL: class up-bioreservoirs-list, win: class window)
 insert BR at the end of the bioreservoir list DBRL:
 if there exists a bioreservoir DBR such that (DBR is a-downstream-bioreservoir-of BR) then
  for DBR = each bioreservoir that is a-downstream-bioreservoir-of BR do
   insert DBR at the end of the bioreservoir list DBRL:
 if there exists a bioprocess DBP such that (DBP is a-down-bioprocess-of BR) then
  for DBP = each bioprocess that is a-down-bioprocess-of BR do
   insert DBP at the end of the bioprocess list DBPL:
   if there exists a bioprocess DSBP such that (DSBP is a-downstream-bioprocess-of DBP) then
    for DSBP = each bioprocess that is a-downstream-bioprocess-of DBP do
     insert DSBP at the end of the bioprocess list DBPL:
 for FDBP = each bioprocess in DBPL do
   allow other processing:
   if there exists a bioreservoir that is an-up-bioreservoir-of FDBP then
    for UBR = each bioreservoir that is an-up-bioreservoir-of FDBP do
     if not (there exists a bioreservoir DBR in DBRL such that (DBR is the same object as UBR)) then
      insert UBR at the end of the bioreservoir list UBRL:
 return:
```

TABLE 195

```
LIST-SIMUL-CALLBACK (button: class button, win: class window, panel: class entry-panel)
 SW = the subworkspace of panel:
 SA = the scroll-area upon SW:
 if not (there exists an uil-message-object that is a-selected-message-of SA) then
  go to no-entry-routine:
 BRN = the related-item of dlg:
 BR = the bioreservoir named by BRN:
 DBRL = the downstream-bioreservoirs-list that is the-downstream-bioreservoirs-list-of BR:
 DBPL = the downstream-bioprocesses-list that is the-downstream-bioprocesses-list-of BR:
 UBRL = the up-bioreservoirs-list that is the-up-bioreservoirs-list-of BR:
 create a bm-downstream-bp-simulation-list CDBPL:
 create a down-br-simulation-list CDBRL:
 create an up-br-simulation-list CUBRL:
 call READ-SCROLL-PROC (DBRL, DBPL, UBRL, CDBRL, CDBPL,CUBRL,win.dlg):
 transfer CDBPL to SW at (395, 70):
 transfer CDBRL to SW at (-35, 69):
 transfer CUBRL to SW at (53, 69):
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then disable de:
 disable button:
 return:
no-entry-routine:
  md = call uil-create-message-dialog ("md", "No bioModel has been selected for this simulation. Click on the ? button for
help, follow the instructions and click on LIST again", win, 0, 0):
  return:
```

TABLE 197

```
ACTIVATE-SIMULATION-PROC (win: class window, panel: class entry-panel)
 SW = the subworkspace of panel:
 DBPL = the bm-downstream-bp-simulation-list upon SW:
 DBRL = the down-br-simulation-list upon SW:
 UBRL = the up-br-simulation-list upon SW:
 M = the model-definition upon SW:
 BCB = the subworkspace of the c-basal-density-source-tracer upon SW:
 User-BC = the user-bc-br-list upon BCB:
 call BM-OF-MODEL-PROC (M):
 call ACTIVATE-MODEL-PROC (DBPL, DBRL, UBRL):
 change the text of the free-text upon BCB to "Sources of values for Basal
    Concentration within: [the text of the items-belonging-to-this-model of M]":
 for DBR = each bioreservoir in DBRL do
    allow other processing:
    PO = the biopool upon the subworkspace of DBR:
    if the basal-density of PO /= 9.9e-99 then insert DBR at the end of the bioreservoir list User-BC
    else call BASAL-DENSITY-PROC (DBR, panel):
 for UBR = each bioreservoir in UBRL do
    allow other processing:
    PO = the biopool upon the subworkspace of UBR:
    if the basal-density of PO /= 9.9e-99 then insert UBR at the end of the bioreservoir list User-BC
    else call BASAL-DENSITY-PROC (UBR, panel):
  if there exists a text-pushbutton st upon SW such that (the id of st = "st") then
    disable st :
 return:
```

TABLE 196

```
READ-SCROLL-PROC (DBRL: class downstream-bioreservoirs-list. DBPL: class downstream-bioprocesses-list. UBRL:
class up-bioreservoirs-list. CDBRL: class down-br-simulation-list. CDBPL: class bm-downstream-bp-simulation-list.
CUBRL: class up-br-simulation-list. win: class window. panel: class entry-panel)
 SW = the subworkspace of panel:
 M = the model-definition upon SW:
 BRN = the related-item of panel:
 ST. SBTM = text:
 n: integer = 0:
 scroll-array = call get-selected-message (panel."select-biomodel-scroll"):
 for ST = each text in scroll-array do
    BMN = symbol (ST):
    BM = the biomodel named by BMN:
    conclude that BM is A-SIMULATED-BM-OF M:
    n = n + 1:
    if n = 1 then SBMT = ST else
    if n > 1 then SBMT = "[SBMT]. [ST]":
    show the SW of BM at three-quarter scale with its top right corner 40 units below the top right corner of the screen:
 for DBR = each bioreservoir in DBRL do
    if DBR is BR-CONTAINED-IN BM then insert DBR at the end of the bioreservoir list CDBRL:
 conclude that the lenght of CDBRL = the number of elements in CDBRL:
 for DBP = each bioprocess in DBPL do
    if DBP is BP-CONTAINED-IN BM then
       insert DBP at the end of the bioprocess list CDBPL:
 conclude that the lenght of CDBPL = the number of elements in CDBPL:
 for UBR = each bioreservoir in UBRL do
    if UBR is BR-CONTAINED-IN BM then insert UBR at the end of the bioreservoir list CUBRL:
 conclude that the lenght of CUBRL = the number of elements in CUBRL:
 change the text of the items-belonging-to-this-model of M to "[SBMT]":
 change the name of CDBPL to symbol ("BPs--downstream-of-[BRN]-within-[SBMT]"):
 change the name of CDBRL to symbol ("BRs-downstream-of-[BRN]-within-[SBMT]"):
 change the name of CUBRL to symbol ("up-BRs-of-[BRN]-within-[SBMT]"):
 conclude that the ref-biomodel of CDBPL = "[SBMT]":
 if there exists a text-pushbutton ac upon SW such that (the id of ac = "ac ") then enable ac.
 return:
```

TABLE 198

```
ACTIVATE-SIMULATION-PROC (win: class window. panel: class entry-panel)
 SW = the subworkspace of panel:
 DBPL = the bm-downstream-bp-simulation-list upon SW:
 DBRL = the down-br-simulation-list upon SW:
 UBRL = the up-br-simulation-list upon SW:
 M = the model-definition upon SW:
 BCB = the subworkspace of the c-basal-density-source-tracer upon SW:
 User-BC = the user-bc-br-list upon BCB:
 call BM-OF-MODEL-PROC (M):
 call ACTIVATE-MODEL-PROC (DBPL. DBRL. UBRL):
 change the text of the free-text upon BCB to "Sources of values for Basal
    Concentration within: [the text of the items-belonging-to-this-model of M]":
 for DBR = each bioreservoir in DBRL do
    allow other processing:
    PO = the biopool upon the subworkspace of DBR:
    if the basal-density of PO /= 9.9e-99 then insert DBR at the end of the bioreservoir list User-BC
    else
       call BASAL-DENSITY-PROC (DBR. panel):
 for UBR = each bioreservoir in UBRL do
    allow other processing:
     PO = the biopool upon the subworkspace of UBR:
    if the basal-density of PO /= 9.9e-99 then insert UBR at the end of the bioreservoir list User-BC
    else
       call BASAL-DENSITY-PROC (UBR. panel):
  if there exists a text-pushbutton st upon SW such that (the id of st ="st ") then disable st :
 return:
```

```
BM-OF-MODEL-PROC. (M: class model-definition)
 if there exists a biomodel BM such that (BM is A-SIMULATED-BM- OF M) then
  for BM = each biomodel that is A-SIMULATED-BM-OF M do
   change the status-color icon-color of BM to brown:
   if there exists a biomodel BM1 such that (BM1 is BM-CONTAINED-IN BM) then
    for BM1 = each biomodel that is BM-CONTAINED-IN BM do
      allow other processing:
      conclude that BM1 is A-SIMULATED-BM-OF M:
      change the status-color icon-color of BM1 to brown:
 return:
```

TABLE 199

```
ACTIVATE-MODEL-PROC (DBPL: class downstream-bp-simulation-list.
DBRL: class down-br-simulation-list. UBRL: class up-br-simulation-list)
 for DBP = each bioprocess in DBPL do
  allow other processing:
  activate the subworkspace of DBP:
  change the status-color icon-color of DBP to brown:
  conclude that the status of DBP is downstream-activated:
 for DBR = each bioreservoir in DBRL do
  allow other processing:
  activate the subworkspace of DBR:
  change the status-color icon-color of DBR to brown:
  conclude that the status of DBR is downstream-activated:
 for UBR = each bioreservoir in UBRL do
  allow other processing:
  exit if the status of UBR is downstream-activated:
  activate the subworkspace of UBR:
  change the status-color icon-color of UBR to brown:
  conclude that the status of UBR is back-acuvated:
 return:
```

TABLE 200

```
BASAL-DENSITY-PROC (BR: class bioreservoir. dlg: class item)
  P() = the biopool upon the SW of BR:
  BC = the basal-density of P():
  top = the top-container that is the-container-of-br BR:
  conclude that the volume of top = the volume of top:
  BCB = the SW of the basal-density-source-tracer upon the SW of dlg:
  if BR is a sol-mol-reservoir then:
    NBC = the normal-basal-concentration of BR:
    if NBC /= 9.9e-99 then
      conclude that the basal-density of P() = NBC * 6.023e23 * the volume of top:
      insert BR at the end of the NBC-bc-br-list upon BCB:
      return:
    else go to main-routine:
  else:
    NBD = the normal-basal-density of BR:
    if NBD /= 9.9e-99 then
      conclude that BC = NBD:
      insert BR at the end of the NBD-bc-br-list upon BCB:
      return:
    else go to main-routine:
main-routine:
  if the scaling-density of P() = 99.999 then:
    insert BR at the end of the no-bc-br-list upon BCB:
    return:
  else:
    AK = the scaling-density of P():
    if the scaled-basal-amount of BR /= 9.9e-99 then:
      conclude that BC = SBA * AK:
      insert BR at the end of the SBA-bc-br-list upon BCB:
      return:
    else:
      case the physiol-abundance of BR is
        highest: conclude that BC = 1.0 * AK:
        abundant: conclude that BC = 0.9 * AK:
        steady-state: conclude that BC = 0.5 * AK:
        low-induced: conclude that BC = 0.1 * AK:
        only-induced: conclude that BC = 1.0e-9 * AK:
      insert BR at the end of the pab-bc-br-list upon BCB:
  return
```

TABLE 201

```
START-CONC-INPUT-CALLBACK (button: class button. win: class window. dlg: class dialog)
  P() = the biopool upon the subworkspace of the bioreservoir named by the related-item of dlg:
  top = the top-container that is the-container-of-br the bioreservoir named by the related-item of dlg:
  define two local float values V and EV. two quantity values t and incr. and one integer value ET:
  set t = the current-simulation-time of dlg:
  set incr = the current-simulation-time-increment of dlg:
  set V equal to the quantity extracted from the value of the edit-box upon dlg which id is "conc-entry-value":
  set EV = V (molar) * 6.023e23 (molecules/liter) * the volume (liters per top-container) of top:
  disable button;
  set ET equal to the quantity extracted from the value of the edit- box upon dlg which id is "entry-time";
  call START-SIMULATION-PROC (win. dlg);
  wait until t >= ET checking every 1 second;
  conclude that the density-entry of P() = EV;
  wait until t >= ET + incr checking every 1 second:
  conclude that the density-entry of P() = 0.0;
  if t > ET + incr then return
```

```
START-DENSITY-INPUT-CALLBACK (button: class button. win: class window. dlg: class dialog)
  P() = the biopool upon the subworkspace of the bioreservoir named by the related-item of dlg:
  define a local float value EV. two quantity values t and incr. and an integer value ET:
  set t = the current-simulation-time of dlg:
  set incr = the current-simulation-time-increment of dlg:
  set EV equal to the quantity extracted from the value of the edit-box upon dlg which id is "density-entry-value".
  disable button:
  set ET equal to the quantity extracted from the value of the edit-box upon dlg which id is "entry-time";
  call START-SIMULATION-PROC (win. dlg):
  wait until t >= ET checking every 1 second:
  conclude that the density-entry of P() = EV;
  wait until t >= ET + incr checking every 1 second;
  conclude that the density-entry of P() = 0.0:
  if t > ET + incr then return
```

```
START-RELATIVE-INPUT-CALLBACK (button: class button. win: class window. dlg: class dialog)
  BR: = the bioreservoir named by the related-item of dlg:
  P() = the biopool upon the subworkspace of the bioreservoir named by the related-item of dlg:
  define a local float EV. two quantities t and incr. and an integer ET:
  set t = the current-simulation-time of dlg:
  set incr = the current-simulation-time-increment of dlg:
  set EV equal to the quantity extracted from the value of the edit-box upon dlg which idis "factor-entry-value":
  disable button:
  set ET equal to the quantity extracted from the value of the edit-box upon dlg which id is "entry-time":
  call START-SIMULATION-PROC (win. dlg);
  wait until t >= ET checking every 1 second:
  conclude that the scaled-entry of P() = EV;
  wait until t >= ET + incr checking every 1 second:
  conclude that the scaled-entry of P() = 0.0;
  if t > ET + incr then return
```

TABLE 202

```
START-PERIODIC-CONC-INPUT-CALLBACK (button: class button. win: class window. dlg: class dialog)
  P() = the biopool upon the subworkspace of the bioreservoir named by the related-item of dlg:
  define two local float s V and EV. two quantities t and incr. and four integers ET. EI. EF. and n:
  set t = the current-simulation-time of dlg:
  set incr = the current-simulation-time-increment of dlg:
  set V equal to the quantity extracted from the value of the edit-box upon dlg which id is "conc-entry-value".
  set EV = V (molar) * 6.023e23 (molecules/liter) * the volume (liters per top-container) of top:
  disable button:
  set ET equal to the quantity extracted from the value of the edit-box upon dlg which id is "entry-time":
  set EF equal to the quantity extracted from the value of the edit-box upon dlg which id is "entry-frequency".
  set EI equal to the quantity extracted from the value of the edit-box upon dlg which id is "entry-interval":
  call START-SIMULATION-PROC (win. dlg);
  wait until t >= ET checking every 1 second:
  conclude that the density-entry of P() = EV;
  wait until t >= ET + incr checking every 1 second:
  conclude that the density-entry of P() = 0.0;
  for n = 1 to EF - 1 do
  wait until t >= ET + EI * n checking every 1 second:
  conclude that the density-entry of P() = EV:
  wait until t >= ET + EI * n + incr checking every 1 second:
  conclude that the density-entry of P() = 0.0:
  if t > ET + EI * n + incr then return;
```

TABLE 203

```
START-SIMULATION-PROC (win: class window, panel: class entry-panel)
 M = the model-definition upon the subworkspace of panel:
 BRN: symbol = the related-item of panel:
 if there exists a text-pushbutton pa upon SW such that (the id of pa = "pa ") then enable pa :
 call start-model (M):
 if the model-simulation-status of M is running then
    sd = call uil-create-message-dialog ("sd", "The simulation starting from\ [BRN] and within
       [the text of the items-belonging-to-this-model of  M]  has started at [the current time as a
       time stamp]", win, 0, 0):
```

TABLE 204

```
MODEL-SIMULATION-PROC ()
 M = call g2-get-current-model-definition ():
 if the item superior to the workspace of M is a entry-panel then
    EP = the entry-panel superior to the workspace of M:
    conclude that the current-simulation-time of EP is equal to the model-
       simulation-time of M:
    conclude that the current-simulation-time-increment of EP is equal to the
       model-simulation-time-increment of M:
 {if attribute-x >= y then
  call g2-set-model-simulation-time (M, n):
  call g2-set-model-simulation-time-increment (M, m):}
 {if attribute-x2 >= y2 then
    Z2, n2, m2 = call cont-model-proc (attribute-x2):
    call g2-set-model-simulation-time (Z2, n2):
    call g2-set-model-simulation-time-increment (Z2, m2):
    call g2-start-model (Z2):}
 start COMPUTE-GRAPH-TRANSFORMS-PROC ():
```

TABLE 206

```
PAUSE-SIMULATION-CALLBACK (button: class uil-button, win: class window,
panel: class entry-panel)
  SW = the workspace of button:
  M = the model-definition upon SW:
  call g2-pause-model (M):
  if there exists a text-pushbutton rm upon SW such that (the id of rm ="rm") then enable rm :
  disable button :
```

```
RESUME-SIMULATION-CALLBACK (button: class uil-button, win: class window,
panel: class entry-panel)
  SW = the workspace of button:
  M = the model-definition upon SW:
  call g2-resume-model (M):
  if there exists a text-pushbutton pa upon SW such that (the id of pa ="pa ") then enable pa :
  disable button :
```

TABLE 205

```
COMPUTE-GRAPH-TRANSFORMS-PROC ()

GT: class graph-tracer:
SW: class kb-workspace:
TF: class transf-factor-var:
T: class graph-transf-var:
C: class variable-or-parameter:

  for each graph-tracer GT do
    if there is a bionode-object BO connected to GT and GT has a SW
  then
  for each transf-factor-var TF upon SW do
    if BO is a biopool then
      case the class of TF of
          scaled-conc-transf-factor:    C = the scaled-amount of BO:
          input-transf-factor :         C = the input-rate of BO:
          output-transf-factor :        C = the output-rate of BO:
          accum-transf-factor :         C = the accumulation-rate of BO:
          density-transf-factor :       C = the density of BO:
    else if  BO is a bioengine and the transf-factor-var upon SW is a
          velocity-transf-factor then   C = the velocity of BO:
    conclude that the transform of TF =
    (if C <= 1.0e-9 then C * 1.0e9 else (if C <= 1.0e-8 then C * 1.0e8 else
    (if C <= 1.0e-7 then C * 1.0e7 else (if C <= 1.0e-6 then C * 1.0e6 else
    (if C <= 1.0e-5 then C * 1.0e5 else (if C <= 1.0e-4 then C * 1.0e4 else
    (if C <= 1.0e-3 then C * 1.0e3 else (if C <= 1.0e-2 then C * 1.0e2 else
    (if C <= 1.0e-1 then C * 1.0e1 else (if C <= 1.0 then C * 1.0 else
    (if C <= 1.0e1 then C * 1.0e-1 else (if C <= 1.0e2 then C * 1.0e-2 else
    (if C <= 1.0e3 then C * 1.0e-3 else (if C <= 1.0e4 then C * 1.0e-4 else
    (if C <= 1.0e5 then C * 1.0e-5 else (if C <= 1.0e6 then C * 1.0e-6 else
    (if C <= 1.0e7 then C * 1.0e-7 else (if C <= 1.0e8 then C * 1.0e-8 else
    (if C <= 1.0e9 then C * 1.0e-9 else (if C <= 1.0e10 then C*1.0e-10else
    (if C <= 1.0e11then C*1.0e-11 else (if C<=1.0e12 then C*1.0e-12 else
    (if C<=1.0e13 then C *1.0e-13 else (if C<=1.0e14 then C*1.0e-14 else
    (if C<=1.0e15 then C*1.0e-15 else (if C<= 1.0e16 then C*1.0e-16 else
    (if C<=1.0e17 then C*1.0e-17 else (if C<=1.0e18 then C*1.0e-18 else
    (if C <= 1.0e19 then C * 1.0e-19 else 1.0e20))))))))))))))))))))))))):
    conclude that TF =
       (if C <= 1.0e-9 then 1.0e9 else (if C <= 1.0e-8 then 1.0e8 else
       (if C <= 1.0e-7 then 1.0e7 else (if C <= 1.0e-6 then 1.0e6 else
       (if C <= 1.0e-5 then 1.0e5 else (if C <= 1.0e-4 then 1.0e4 else
       (if C <= 1.0e-3 then 1.0e3 else (if C <= 1.0e-2 then 1.0e2 else
       (if C  <= 1.0e-1 then 1.0e1 else (if C <= 1.0 then 1.0 else
       (if C <= 1.0e1 then 1.0e-1 else (if C <= 1.0e2 then 1.0e-2 else
       (if C <= 1.0e3 then 1.0e-3 else (if C <= 1.0e4 then 1.0e-4 else
       (if C <= 1.0e5 then 1.0e-5 else (if C <= 1.0e6 then 1.0e-6 else
       (if C <= 1.0e7 then 1.0e-7 else (if C <= 1.0e8 then 1.0e-8 else
       (if C <= 1.0e9 then 1.0e-9 else (if C <= 1.0e10 then 1.0e-10 else
       (if C <= 1.0e11 then 1.0e-11 else (if C <= 1.0e12 then 1.0e-12 else
       (if C <= 1.0e13 then 1.0e-13 else (if C <= 1.0e14 then 1.0e-14 else
       (if C <= 1.0e15 then 1.0e-15 else (if C <= 1.0e16 then 1.0e-16 else
       (if C <= 1.0e17 then 1.0e-17 else (if C <= 1.0e18 then 1.0e-18 else
       (if C <= 1.0e19 then 1.0e-19 else 1.0e20))))))))))))))))))))))))
```

TABLE 207

	an user-menu-choice
Label	show-s.graph
Applicable class	bioreservoir
Condition	the subworkspace of the scaled-br-graph-tracer upon the subworkspace of the item does not exist
Action	in order start SCALED-BR-TRANSF-GRAPH-PROC (the scaled-br-graph-tracer upon the subworkspace of the item, this window) and create a min-rev-display T and change the text of T to "G" and transfer T to the workspace of the item at (middle-x (the item) - 29, middle-y (the item) + 6)

```
SCALED-BR-GRAPH-PROC (tracer: class scaled-br-graph-tracer, win: class window )
  BR = the bioreservoir superior to the workspace of tracer;
  if the subworkspace of tracer exists then
    show the subworkspace of tracer on Win with its top right corner 35 units below the top right
      corner of the screen
  else
     create a workspace SW by cloning the subworkspace of SCALED-BR-TRANSF-GRAPH-MASTER;
     G = the graph upon SW;
     change the text of the label-to-display of G to "[the name of BR]";
     change the size of SW to minimum;
     show SW on Win with its top right corner 35 units below the top right corner of the screen;
     make SW the subworkspace of tracer;
     for TF = each transf-factor-var upon SW do
        change the text of the ref-variable of TF to "[the name of BR]";
```

TABLE 208

	a velocity-transf-factor
Options	do not forward chain, breadth first backward chain
Names	none
Tracing and breakpoints	default
Data type	float
Initial value	0.0
Last recorded value	0.0, valid indefinitely
History keeping spec points = 3	keep history with maximum number of data
Validity interval	supplied
Formula	none
Simulation details	no simulation formula yet
Initial value for simulation	default
Data server	inference engine
Default update interval	3 seconds
Transform	0.0
Ref variable	none
Label	velocity
X	x

TABLE 209

	an user-menu-choice
Label	hide-s.graph
Applicable class	bioreservoir
Condition	the subworkspace of the scaled-br-graph-tracer upon the subworkspace of the item exists
Action	in order delete the subworkspace of the scaled-br-graph-tracer upon the subworkspace of the item and delete the min-rev-display nearest to the item

TABLE 210

```
RESET-SIMULATION-CALLBACK (button: class uil-button. win: class window. panel: class
entry-panel)
 M = the model-definition upon the workspace of button:
 call g2-reset-model (M):
 conclude that the current-simulation-time of dlg = 0.0:
 conclude that the current-simulation-time-increment of dlg = 0.0:
 call RESET-PROC (M. dlg):
 allow other processing:
 if there exists a text-pushbutton st upon SW such that (the id of st = "st ") then disable st :
 if there exists a text-pushbutton pa upon SW such that (the id of pa = "pa") then disable pa:
 if there exists a text-pushbutton rm upon SW such that (the id of rm = "rm") then disable rm:
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then disable de:
```

```
RESET-PROC (M: class model-definition. panel: class entry-panel)
 SW = the subworkspace of panel:
 if there exists a biomodel that is A-SIMULATED-BM-OF M then
  for BM = each biomodel that is A-SIMULATED-BM-OF M do
    allow other processing:
    change the status-color icon-color of BM to medium-aquamarine:
    conclude that BM is not A-SIMULATED-BM-OF M:
 if there exists a bm-downstream-bp-simulation-list CDBPL upon SW then
    for DBP = each bioprocess in CDBPL do
       allow other processing:
       change the status-color icon-color of DBP to slate-blue:
       conclude that the status of DBP is available:
       deactivate the subworkspace of DBP:
    delete DBPL:
 if there exists a down-br-simulation-list CDBRL upon SW then
    for DBR = each bioreservoir in CDBRL do
       allow other processing:
       deactivate the subworkspace of DBR:
       change the status-color icon-color of DBR to medium-aquamarine:
       conclude that the status of DBR is available:
       delete CDBRL:
 if there exists a up-br-simulation-list UBRL upon SW then
    for UBR = each bioreservoir in UBRL do
       allow other processing:
       deactivate the subworkspace of UBR:
       change the status-color icon-color of UBR to medium-aquamarine:
       conclude that the status of UBR is available:
    delete UBRL:
 allow other processing:
 if there exists a text-pushbutton ls upon SW such that (the id of ls = "ls") then enable ls:
 if there exists a text-pushbutton ac upon SW such that (the id of ac = "ac") then disable ac:
 return
```

TABLE 211

```
ALT-CREATE-CONC-INPUT-PANEL-PROC (BR: class bioreservoir, win : class window) = (class entry-panel )
 panel = call clone-panel (MASTER-ALT-CONCENTRATION-ENTRY-PANEL):
 SW = the subworkspace of panel:
 change the text of the user-restrictions of SW to "unless in uil-build mode:
    menu choices for workspace include: hide-workspace, shrink-wrap:
    non-menu choices for kb-workspace include: scale-workspace, show-workspace,
      move-workspace, move-workspaces-beyond-window-margin:
    attributes visible for kb-workspace include: nothing":
 create a panel-text FT:
 transfer FT to SW at (-20, 300):
 change the text of the id of panel to "[BRN]-conc-entry-panel":
 change the text of FT to "1. CONCENTRATION INPUT FOR: [the name of BR]":
 st = call create-pushbutton ("st", the symbol ALT-START-CONC-INPUT-CALLBACK,"START", panel, 138, 60):
 disable st:
 return panel:
```

TABLE 212

```
ALT-LIST-SIMUL-CALLBACK (button: class button, win: class window, panel : class entry-panel )
 SA = the uil-scroll-area upon the subworkspace of dlg:
 BR = the bioreservoir named by the related-item of dlg:
 BRL= the downstream-bioreservoirs-list that is the-downstream-bioreservoirs-list-of BR:
 BPL = the downstream-bioprocesses-list that is the-downstream-bioprocesses-list-of BR:
 if not (there exists an uil-message-object that is a-selected-message-of SA) then go to no-entry-routine:
 create a bm-downstream-bp-simulation-list DBPL:
 create a down-br-simulation-list DBRL:
 create an up-br-simulation-list UBRL:
 call ALT-READ-SCROLL-PROC (BPL, BRL, DBPL, DBRL, UBRL, in, dlg):
 transfer DBPL to the subworkspace of dlgat (395, 70):
 transfer DBRL to the subworkspace of dlgat (-35, 69):
 transfer UBRL to the subworkspace of dlgat (53, 69):
 if there exists a text-pushbutton de upon the subworkspace of dlgsuch that (the id of de = "de") then disable de:
 disable button:
 return:
no-entry-routine:
   md = call create-message-dialog ("md", "No bioModel has been selected for this simulation.Click on the ? button
        for help, follow the instructions and click on LIST again", win, 0, 0):
   return:
```

TABLE 213

```
ALT-READ-SCROLL-PROC (BPL: class downstream-bioprocesses-list, BRL: class downstream-bioreservoirs-lst, DBPL: class
bm-downstream-bp-simulation-list, DBRL: class down-br-simulation-list, UBRL: class up-br-simulation-list, win: class window,
panel: class entry-panel)
  scroll-array = call get-selected-message (panel, "select-biomodel-scroll"):
  for ST = each text in scroll-array do
   BMN = symbol (ST):
   BM = the biomodel named by BMN:
   conclude that BM is A-SIMULATED-BM-OF M
   n = n + 1:
   if n = 1 then SBMT = ST else
   if n > 1 then SBMT = "[SBMT], [ST]":
   show the subworkspace of BM at three-quarter scale with its top right corner 40 units below the top right corner of the screen.
  for DBR = each bioreservoir in BRL do
   allow other processing:
   if DBR is BR-CONTAINED-IN BM then insert DBR at the end of the bioreservour list DBRL:
  conclude that the lenght of DBRL = the number of elements in DBRL:
  for DBP = each bioprocess in BPL do
   allow other processing:
   if DBP is BP-CONTAINED-IN BM then insert DBP at the end of the bioprocess list DBPL:
  conclude that the lenght of DBPL = the number of elements in DBPL:
  for FDBP = each bioprocess in DBPL do
   allow other processing:
   if there exists a bioreservoir that is an-up-bioreservoir-of FDBP then
     for UBR = each bioreservoir that is an-up-bioreservoir-of FDBP do
       if not (there exists a bioreservoir DBR in DBRL such that (DBR is the same object as UBR)) then
         insert UBR at the end of the bioreservoir list UBRL:
  conclude that the lenght of UBRL = the number of elements in UBRL:
  change the name of DBPL to symbol ("BPs-downstream-of-[the related-item of panel]-within-[SBMT]"):
  change the name of DBRL to symbol("BRs-downstream-of-[the related-item of panel]-within-[SBMT]"):
  change the name of UBRL to symbol("up-BRs-of-[the related-item of panel]-within-[SBMT]"):
  conclude that the ref-biomodel of DBPL = "[SBMT]":
  if there exists a text-pushbutton dr upon SW such that (the id of dr ="dr") then enable dr:
  return:
```

TABLE 214

```
SIMUL-DRAW-DBP-CALLBACK (button: class button. win: class item. panel: class entry-panel)
 WS = the subworkspace of panel:
 DBPL = the bm-downstream-bp-simulation-list upon WS:
 BR = the bioreservoir named by the related-item of panel:
 if there exists a bioprocess that is a-down-bioprocess-of BR then
   if there exists a c-neural-path-tracer upon WS then
   for NT = each c-neural-path-tracer upon WS do
     delete NT:
   create a c-neural-path-tracer NT by cloning master-neuro-pathway:
   transfer NT to WS at (510. 290):
   SW = the subworkspace of NT:
   BT = the biomodel-title upon SW:
   conclude that the label of BT = "[the label of BT]: [the label of BR]":
   show SW at three-quarter scale with its top left corner 50 units below the top left corner of the screen:
   call CONFIGURE-PATH-TRACER-PROC (NT. win):
   x = the x-pos upon SW:  conclude that x = -300:
   y = the y-pos upon SW:  conclude that y = 20:
   create a bioreservoir LBR by cloning BR:  activate the subworkspace of LBR:
   PO = the biopool upon the subworkspace of LBR:
   create a panel-text TI:
   change the text of TI to "[the master-bioreservoir of LBR]":
   transfer TI to BRW at (middle-x (PO). middle-y (PO) - 80):
   transfer LBR to SW at (x. y):
   conclude that y = y - 40:
   call SIMUL-CREATE-LOCAL-BP-PROC (BR. LBR. WS. SW. win):
   for NBP = each bioprocess upon SW do
     MBP = the bioprocess named by the master-bioprocess of NBP:
     if there exists a bioreservoir that is an-up-bioreservoir-of MBP then
       call SIMUL-CREATE-UP-LOCAL-BR-PROC (NBP. MBP. WS. SW. win):
 call ALT-ACTIVATE-SIMULATION-PROC (panel. win):
 change the size of SW to minimum:
 disable button :
 if there exists a text-pushbutton st upon SW such that (the id of st = "st") then disable s:
 return:
```

TABLE 215

```
SIMUL-CREATE-LOCAL-BP-PROC (BR: class bioreservoir. LBR: class
 bioreservoir. WS: class workspace. SW: class workspace. win: class window)
 DBPL = the bm-downstream-bp-simulation-list upon WS:
 x = the x-pos upon SW:     y = the y-pos upon SW:
 if there exists a bioprocess that is a-down-bioprocess-of BR then
  for DBP = each bioprocess that is a-down-bioprocess-of BR do
   if DBP is a member of the bioprocess list DBPL then
     if there exists a bioprocess MBP upon SW such that (the master-bioprocess of MBP =
         the master-bioprocess of DBP ) then
       LDBP = MBP:
       go to connect-BP-routine:
     else
       conclude that x = x + 110:
       create a bioprocess LDBP by cloning DBP:
       activate the subworkspace of LDBP:
       transfer LDBP to SW at (x. y):
       r = random (3. 35):    t = random (20. 76):
       create a connection connected between LBR newly locating it at right at the position given by r and LDBP newly
         locating it at top at the position given by t with direction output:
       conclude that y = y - 40:
       call SIMUL-CREATE-LOCAL-BR-PROC (DBP. LDBP. WS.  SW. win):
  return:
connect-BP-routine:
  for C = each connection connected to LBR do
    if there exists a connection DC connected to LDBP such that (DC is the same object as C) then return:
  t = random (20. 76):    r = random (3. 35):
  p = x-rel-position (LBR. LDBP):
  if p > 0 then
    create a connection of class c-line connected between LBR newly locating it at right at the position given by r and LDBP
        newly locating it at bottom at the position given by t with direction output. with vertices 6 -60 else
  if p < 0 then
    create a connection of class c-line connected between LBR newly locating it at bottom at the position given by t and
        LDBP newly locating it at left 52 with direction output. with vertices -60:
  call SIMUL-CREATE-LOCAL-BR-PROC (DBP. LDBP. WS. SW.win):
  return:
```

TABLE 216

```
SIMUL-CREATE-LOCAL-BR-PROC (BP: class bioprocess. LBP: class
bioprocess. WS: class workspace. SW: class workspace. win: class window)

 DBRL = the down-br-simulation-list upon WS:
 x = the x-pos upon SW:
 y = the y-pos upon SW:
 if there exists a bioreservoir that is a-down-bioreservoir-of BP then
  for DBR = each bioreservoir that is a-down-bioreservoir-of BP do
   if DBR is a member of the bioreservoir list DBRL then
    if there exists a bioreservoir MBR upon SW such that (the master-
        bioreservoir of MBR = the master-bioreservoir of DBR) then
      LDBR = MBR:
      go to connect-BR-routine:
    else
      allow other processing:
      conclude that x = x + 110:
      if x > 2400 then show SW at one-quarter scale with its top left corner
          50 units below the top left corner of the screen else
      if x > 1400 and x < 2400 then show SW at half scale with its top left
          corner 50 units below the top left corner of the screen:
      create a bioreservoir LDBR by cloning DBR:
      BRW = the subworkspace of LDBR:
      PO = the biopool upon BRW:
      create a panel-text TI:
      change the text of TI to "[the master-bioreservoir of LDBR]":
      transfer TI to BRW at (middle-x (PO), middle-y (PO) - 80):
      change the size of BRW to minimum:
      transfer LDBR to SW at (x, y):
      r = random (3, 35):
      t = random (20, 76):
      create a connection connected between LBP newly locating it at right
    at the position given by r and LDBR newly locating it at top at the
      position given by t with direction output:
      conclude that y = y - 40:
      call SIMUL-CREATE-LOCAL-BP-PROC (DBR. LDBR. WS. SW.
        win):
  return:

connect-BR-routine:
   for C = each connection connected to LBP do
    if there exists a connection DC connected to LDBR such that (DC is the
       same object as C) then return:
   t = random (20, 76):
   r = random (3, 35):
   p = x-rel-position (LBP. LDBR):
   if p > 0 then
     create a connection of class e-line connected between LBP newly
       locating it at right at the position given by r and LDBR newly
     locating it at bottom at the position given by t with direction output.
     with vertices 6 -60 else
   if p < 0 then
     create a connection of class e-line connected between LBP newly
   locating it at bottom at the position given by t and LDBR newly
   locating it at left 52 with direction output. with vertices -60:
   call SIMUL-CREATE-LOCAL-BP-PROC (DBR. LDBR. WS. SW. win):
   return
```

TABLE 217

```
SIMUL-CREATE-UP-LOCAL-BR-PROC (NBP: class bioprocess . MBP: class bioprocess. WS:
class workspace. SW: class workspace. win: class window)

 UBRL = the up-br-simulation-list upon WS:
 x = the x-pos upon SW:
 y = the y-pos upon SW:
 conclude that x = the item-x-position of NBP:
 conclude that Y = the item-y-position of NBP + 200:
 for UBR = each bioreservoir that is an-up-bioreservoir-of MBP do
  if UBR is a member of the bioreservoir list UBRL then
   if there exists a bioreservoir MBR upon SW such that (the master-
        bioreservoir of MBR = the master-bioreservoir of UBR) then
     LUBR = MBR:
     go to connect-BR-routine:
   else
     allow other processing:
     conclude that x = x + 110:
     create a bioreservoir LUBR by cloning UBR:
     activate the subworkspace of LUBR:
     BRW = the subworkspace of LUBR:
     PO = the biopool upon BRW:
     create a panel-text TI:
     change the text of TI to "[the master-bioreservoir of LUBR]":
     transfer TI to BRW at (middle-x (PO), middle-y (PO) - 80):
     change the size of BRW to minimum:
     transfer LUBR to SW at (x, y):
     r = random (3, 35):
     t = random (20, 76):
     create a connection connected between NBP newly locating it at bottom at the position given by
        t and LUBR newly locating it at top at the position given by t with direction output:
     conclude that y = y - 40
  return:
connect-BR-routine:
  for C = each connection connected to NBP do
   if there exists a connection DC connected to LUBR such that (DC is the same object as C) then return:
  t = random (20, 76):
  r = random (3, 35):
  p = x-rel-position (NBP, LUBR):
  if p > 0 then
   create a connection of class c-line connected between NBP newly locating it at right at the
      position given by r and LUBR newly locating it at bottom at the position given by t with
      direction output. with vertices 6 -60 else
  if p < 0 then
   create a connection of class c-line connected between NBP newly locating it at bottom at the
position given by t and LUBR newly locating it at left 52 with direction output. with vertices -60:
  return:
```

TABLE 218a

THE-BIOPOOL-OF-R	
First class	biopool
Second class	bioreactant
Relation name	the-biopool-of-R
Inverse of relation	a-bioreactant-of
Type of relation	one-to-many
Relation is symmetric?	no
THE-BIOPOOL-OF-P	
First class	biopool
Second class	bioproduct
Relation name	the-biopool-of-P
Inverse of relation	a-bioproduct-of
Type of relation	one-to-many
Relation is symmetric?	no

TABLE 218b

```
ALT-ACTIVATE-SIMULATION-PROC (panel: class entry-panel, win: class window)
 WS = the subworkspace of panel:
 NT = the neural-path-tracer upon WS:
 SW = the subworkspace of NT:
 M = the model-definition upon WS:
 BCB = the subworkspace of the basal-density-source-tracer upon WS:
 User-BC  = the user-bc-br-list upon BCB:
 simul-n = call uil-unique-id ("simulation-"):
 change the name of SW to symbol("[simul-n]"):
 change the text of the items-belonging-to-this-model of M to "[the name of  SW]":
 for SBR  = each bioreservoir upon SW do
  allow other processing:
  PO = the biopool upon the subworkspace of SBR:
  for PP = each biopool-post upon the subworkspace of SBR do
   for SBP  = each bioprocess upon SW do
    if the master-bioprocess of SBP = the ref-bioprocess of PP then
     for RP = each biorole-post upon the subworkspace of SBP do
      if the ref-bioreservoir of RP = the master-bioreservoir of SBR then RO = the biorole-object connected to RP:
        if RO is a bioreactant then  conclude that PO is THE-BIOPOOL-OF-R RO
        else conclude that PO is THE-BIOPOOL-OF-P RO:
  if the basal-density of PO /= 9.9e-99 then  insert SBR at the end of the bioreservoir list User-BC
  else
    call BASAL-DENSITY-PROC (SBR, panel):
 change the text of the c-panel-text upon BCB to "Sources of values for Basal Concentration within the scope
    of the current simulation:".
 return:
```

TABLE 219

```
ALT-START-CONC-INPUT-CALLBACK (button: class button, win: class  window, panel: class entry-panel)
he related-item of panel:  BR = the bioreservoir named by he related-item of panel::
  PO = the biopool upon the subworkspace of BR:
  top = the top-container that is the-container-of-br BR:
  D = the density-entry of PO:
  incr = the CURRENT-SIMULATION-TIME-INCREMENT of panel:
  t = the CURRENT-SIMULATION-TIME of panel:
  C = call get-edit-box-value(panel, "conc-entry-value"):
  if C = 0.0 then go to no-value-routine
  else go to main-routine:
no-value-routine:
    fd, o, ap, ca = call create-query-dialog ("fd", the symbol none, "No value for the concentration of [the related-item
      of panel] has been entered.  If you want  to run the simulation with no external input then press NONE.  If you
      want an input at  time 0, enter the desired value in the box below and press INPUT.  Otherwise press cancel and
      enter on the panel and press START again.", "0.0", win, 0, 0):
    conclude that the id of o = "input":
    conclude that the label of o = "INPUT":
    conclude that the id of ap = "none":
    conclude that the label of ap = "NONE":
    conclude that the id of ca = "cancel":
    option = call wait-for-dialog (fd, win):
    case option is
      "input":
          C = call uil-get-query-dialog-value (fd):
         go to main-routine:
      "none":
          C = 0.0:
         go to main-routine:
      "cancel":
          return:
  main-routine:
    EV = C [molar] * 6.023e23 [molecules/liter] * the volume [liters per top-container] of top:
    disable button:
    ETtext = call get-edit-box-value(panel, "entry-time"):
    ET = quantity (ETtext):
    allow other processing:
    call START-SIMULATION-PROC (win, panel):
    wait until t >= ET checking every 1 second:
    conclude that DE = EV:
    wait until t >= ET + incr checking every 1 second:
    conclude that DE = 0.0:
    if t > ET + incr then return:
```

TABLE 220

```
ALT-RESET-SIMULATION-CALLBACK (button: class button, win: class window, panel: class entry-panel)

 M = the model-definition upon the workspace of button;
 call g2-reset-model (M);
 conclude that the current-simulation-time of panel = 0.0;
 conclude that the current-simulation-time-increment of panel = 0.0;
 if there exists a biomodel that is a-simulated-bm-of M then
   for BM = each biomodel that is a-simulated-bm-of M do
     allow other processing;
     change the status-color icon-color of BM to medium-aquamarine;
     conclude that BM is not A-SIMULATED-BM-OF M;
 if there exists a bm-downstream-bp-simulation-list DBPL upon SW then delete DBPL;
 if there exists a down-br-simulation-list DBRL upon SW then delete DBRL;
 if there exists a up-br-simulation-list UBRL upon SW then delete UBRL;
 if there exists a text-pushbutton ls upon SW such that (the id of ls = "ls") then disable ls;
 if there exists a text-pushbutton dr upon SW such that (the id of dr = "dr") then disable dr;
 if there exists a text-pushbutton st upon SW such that (the id of st = "st") then disable st;
 if there exists a text-pushbutton pa upon SW such that (the id of pa = "pa") then disable pa;
 if there exists a text-pushbutton rm upon SW such that (the id of rm = "rm") then disable rm;
 if there exists a text-pushbutton de upon SW such that (the id of de = "de") then enable de;
```

TABLE 221

an user-menu-choice

Label	show-bp
Applicable class	biopool-post
Condition	the ref-bioprocess of the item has a value
Action	start POOLPOST-SHOW-BP-PROC (the item, this window)

```
POOLPOST-SHOW-BP-PROC (post: class biopool-post, win: class window)

 if post has a name then BP = the bioprocess named by the ref-bioprocess of post
 else
 if there exists a bioprocess ABP such that (ABP has no name and the master-bioprocess of ABP has a
    current value and the master-bioprocess of ABP = the ref-bioprocess of post) then
  BP = ABP;
 change the size of the subworkspace of BP to minimum;
 case (the toggle-state of post ) of
   show :
     change the color-pattern of post so that light-color is dim-gray , dark-color is light-gray , white-color =
        the symbol black;
     conclude that the toggle-state of post is hide;
     case (the class of post) of
       biopool-p-post:
         show the subworkspace of BP on W at full scale with its center 280 units above the center of the screen;
       biopool-r-post:
         show the subworkspace of BP on W at full scale with its center 280 units below the center of the screen;
   hide:
     change the color-pattern of post so that light-color is light-gray , dark-color is dim-gray , white-color =
        the symbol white;
     conclude that the toggle-state of post is show;
     hide the subworkspace of BP on W;
```

TABLE 222

Class name	experiment-selection
Superior class	biotool
Attributes specific to class	label is ""; ref-bioreservoir; amount-entry is 0.0; factor-entry is 0.0; time-entry is 0 seconds; currently-selected has values true or false (default is FALSE)
Class restrictions	unless in developer, or modeler mode: selecting any experiment-selection implies table
Attribute displays	label offset by (-37, 12)
Icon description	experiment-selection-pattern (width 70; height 40; dark-color , light-color , type-color , white-color

TABLE 223

```
BR-SCROLL-CALLBACK (tracer: class br-scroll-tracer, win: class window, dlg: class entry-panel)
  create a workspace display by cloning the SW of BR-SCROLL;
  make display the SW of tracer;
  create a text-list BRNL;
  for each bioreservoir BR with a name do
     insert "<the name of BR>" at the end of the text list BRNL;
  add-each-message-in-list-BRNL to the scroll-area upon display;
  change the size of display to minimum;
  show display on win at the bottom right corner of the screen;
  delete BRNL;
  return
```

TABLE 224

Class name	process-button
Superior class	action-button
Attributes specific to class	action-proc-name is PROCESS-EXPER-CALLBACK
Icon description	process-button-pattern (width 80; height 30)

```
PROCESS-EXPER-CALLBACK (button: class pushbutton, win: class window, panel: class entry-panel)
  SW = the subworkspace of panel;
  M = the model-definition upon SW;
  create a downstream-bioreservoirs-list BRL;
  transfer BRL to SW at (-370, -335);
  create a experiment-selections-list SL;
  transfer SL to SW at (-275, -335);
  create a downstream-bioprocesses-list BPL;
  transfer BPL to SW at (-180, -335);
  create a scroll-text-list PSML;
  for ES = each experiment-selection upon SW do
    allow other processing;
    if the text of the ref-bioreservoir of ES /= "none" and the text of the currently-selected of ES = "true" then
       insert ES at the end of the experiment-selection list SL;
       BR = the bioreservoir named by the ref-bioreservoir of ES;
       call FILL-BR-DOWNSTREAM-LISTS-PROC (BR, BRL, BPL, win);
  for DBP = each bioprocess in BPL do
    for DBM = each biomodel that is the-container-of-bp DBP do
      if DBM is not a bioreservoir-bin then
         insert "[the name of DBM]" at the end of the text list PSML;
  for DBP = each bioprocess in BPL do
    for DBMT = each text in PSML do
      DBMN = symbol(DBMT);
      DBM = the biomodel named by DBMN;
      if DBM is not the-container-of-bp DBP then remove DBMT from the text list PSML;
  for DBMT = each text in PSML do
     DBMN = symbol(DBMT);
     DBM = the biomodel named by DBMN;
  sa = call create-scroll-area-from-list ("select-biomodel-scroll", 5, PSML, panel, -270, -228);
  change the text of the id of sa to select-biomodel-scroll;
  conclude that the allow-unselect-on-selected-message of sa is true;
  conclude that the allow-multiple-simultaneous-selections of sa is true;
  change the text of the message-selection-method of sa to MESSAGE-SELECTION;
  change the text of the message-unselection-method of sa to MESSAGE-UNSELECTION;
  if there exists a text-pushbutton ls upon SW such that (the id of ls = "ls") then disable ls;
  if there exists a text-pushbutton cl upon SW such that (the id of cl = "cl") then enable cl;
  disable button;.
```

TABLE 225

```
CLEAR-EXPER-CALLBACK (button: pushbutton, win: class window, panel: class entry-panel)
 SW = the subworkspace of panel:
 if there exists an g2-list upon panel then
  for L = each g2-list upon panel do
   delete L:
 if there exists a uil-scroll-area sa upon panel such that (the id of de = "select-biomodel-scroll") then delete sa:
 if there exists a text-pushbutton pr upon SW such that (the id of pr = "pr") then enable pr:
 disable button:
```

TABLE 226

```
LIST-EXPER-CALLBACK (button: class pushbutton, win: class window, panel: class entry-panel)
 SW = the subworkspace of panel:
 M: class model-definition = the model-definition upon SW:
 DBRL = the downstream-bioreservoirs-list upon SW;
 DBPL= the downstream-bioprocesses-list upon SW:
 UBRL = the up-bioreservoirs-list upon SW;
 n: integer = 0;
 if there exists a scroll-area sa upon SW such that (the id of sa = "sa") then
  if not (there exists an message-object that is a-selected-message-of sa) then
   go to no-entry-routine
  else
   create a bm-downstream-bp-simulation-list CDBPL:
   create a down-br-simulation-list CDBRL:
   create an up-br-simulation-list CUBRL:
   call READ-SCROLL-PROC (DBRL, DBPL, UBRL, CDBRL,
       CDBPL, CUBRL, win, panel):
   transfer CDBPL to SW at (-90, -350):
   transfer CDBRL to SW at (-35, -350):
   transfer CUBRL to SW at (20, -350);
 if there exists a text-pushbutton hi upon SW such that (the id of hi = "hi") then disable hi;
 disable button:
 return:
no-entry-routine:
  md = call uil-create-message-dialog ("md", "No bioModel has been selected for this experiment.
   Click on the '? button for help, follow the instructions and click on LIST again", win, 0, 0:.
  return:
```

TABLE 229

```
ACTIVATE-EXPER-CALLBACK (button: class pushbutton, win: class window, panel: class entry-panel)
 WS = the subworkspace of panel:
 ESL= the experiment-selections-list upon WS:
 DBPL= the bm-downstream-bp-simulation-list upon WS;
 val = local text:
 if the user-mode of win is simulation then
  for ES = each experiment-selection in ESL do
   if the amount-entry of ES = 0.0 and the factor-entry of ES = 0.0 then
    go to no-value-routine
   else
    go to main-routine:
no-value-routine:
  fd = call create-query-dialog ("fd", "No value for the amount-entry or factor-entry of [the label of ES] has
      been entered. If you want to run the simulation with no external input then enter YES in the box below
      and press OK. Otherwise, press OK, enter the desired value for this experiment-selection, and press
      ACTIVATE again.", the symbol large, "NO", win, 0, 0);
  val = call get-query-dialog-value (fd):
  if val = "YES" then go to main-routine else return:
main-routine:
  if the number of elements in DBPL = 0 then go to no-bp-routine
  else
   call ACTIVATE-SIMULATION-PROC (win, panel):
   disable button:
  return:
no-bp-routine:
  ad = call create-message-dialog ("ad", "There are no bioProcesses that can be selected for activation", win, 0, 0);
  return:
```

TABLE 227

```
DRAW-EXPER-PATHWAY-CALLBACK (button: class button, win: class window, panel: class entry-panel)
 WS = the subworkspace of panel:
 EL = the experiment-selections-list upon WS:
 DBPL = the bm-downstream-bp-simulation-list upon WS:
 init-x: integer = - 50:
 if there exists a c-navig-path-tracer upon WS then
   for NT = each c-navig-path-tracer upon WS do
     delete NT:
 create a navig-path-tracer NT by cloning MASTER-NAVIG-PATHWAY:
 transfer NT to WS at (30, -240):
 SW = the subworkspace of NT:
 x = the x-pos upon SW:
 y = the y-pos upon SW:
 conclude that y = 120:
 show WS at half scale with its bottom right corner at the bottom right corner of the screen:
 show SW at three-quarter scale with its top left corner 50 units below the top left corner of the screen:
 call CONFIGURE-PATH-TRACER-PROC (NT, win):
 for E = each experiment-selection in EL do
  allow other processing:
  n = n + 1:
  conclude that x = init-x + 110 * (n - 1):
  conclude that y = y - 100:
  BR = the bioreservoir named by the ref-bioreservoir of E:
  if there exists a bioprocess that is a-down-bioprocess-of BR then
     conclude that the status of BR is selected-for-input:
     BT = the biomodel-title upon SW:
     conclude that the label of BT = "[the label of BT]: [the label of BR]":
     create a bioreservoir LBR by cloning BR:
     transfer LBR to SW at (x, y):
     conclude that y = y - 40:
     for DBP = each bioprocess that is a-down-bioprocess-of BR do
      allow other processing:
      conclude that x = x + 110:
      create a bioprocess LDBP by cloning DBP:
      transfer LDBP to SW at (x, y):
      r = random (3, 35):
      t = random (20, 76):
      create a connection connected between LBR newly locating it at right at the position given by r and
        LDBP newly locating it at top at the position given by t with direction output:
      conclude that y = y - 40:
      call CREATE-LOCAL-EXPER-BP-PROC (DBP, LDBP, SW, win):
 change the size of SW to minimum:
 disable button:.
 return
```

TABLE 230

```
START-EXPER-CALLBACK (button: class pushbutton, win: class window, dlg: class entry-panel)
  panel = the SW of dlg:
  call SET-BASAL-DENSITIES-PROC (win, dlg): ESL = the experiment-selections-list upon panel:
  define three local quantities: EV, FE, and ET:
  M = the model-definition upon panel:
  for ES = each experiment-selection in ESL do
     BR = the bioreservoir named by the ref-bioreservoir of ES:
     EV = the amount-entry of ES:
     FE = the factor-entry of ES:
     ET = the time-entry of ES:
     if EV /= 0.0 then
       if BR is a sol-mol-reservoir or BR is a exp-cell-reservoir then
         start CONC-INPUT-PROC (BR, EV, ET, win, dlg)
       else start DENSITY-INPUT-PROC (BR, EV, ET, win, dlg)
     else start DENSITY-INPUT-PROC (BR, FE, ET, win, dlg):
  start running the model M, conclude that the model-simulation-status of M is running, and inform the
    operator that " The simulation within <the items-belonging-to-this-model of M> has started at <the current
    time as a time stamp>":
  enable the button upon panel which id is "pa":
  disable button:
  return
```

TABLE 228

```
CREATE-LOCAL-EXPER-BP-PROC (BP: class bioprocess. LBP: class bioprocess. display: class kb-workspace.
    win: class G2-window)
  define four local integers: p. l. r. and t: :
  x = the x-pos upon display:      y= the y-pos upon display:
  for each bioreservoir BR that is a-down-bioreservoir-of BP do
    for each bioprocess DBP that is a-down-bioprocess-of BR do
      if there is a bioprocess MBP upon SW such that the master-
        bioprocess of MBP is equal to the master-bioprocess of DBP then
        go to connect-BP-routine:
      else:
        conclude that x = x + 110:
        create a bioprocess LDBP by cloning DBP:
        transfer LDBP to display at (x. y):
        create a connection connected between LBP newly locating it at right at the position given by random
          (3. 35) and LDBP newly locating it at top at the position given by random (20. 76) with direction output:
        conclude that y = y - 40:
        call CREATE-LOCAL-EXPER-BP-PROC (DBP. LDBP. display. win)
return:
connect-BP-routine:
  for each connection C connected to LBP do
    if there exists a connection DC connected to MBP such that DC is the same object as C then return:
  if the x-position of LBP is lower than the x-position of MBP) then
    create a connection of class c-line connected between LBP newly locating it at right at the position given by
      random (3. 35) and MBP newly locating it at bottom at the position given by random (20. 76) with direction
      output. with vertices 6 -60 else
  if the x-position of LBP is higher than the x-position of MBP) then
    create a connection of class c-line connected between LBP newly locating it at bottom at the position given
      by (20. 76) and MBP newly locating it at left 52 with direction output. with vertices -60:
  call CREATE-LOCAL-EXPER-BP-PROC (DBP. MBP. display. win):
return
```

TABLE 231

```
CONC-INPUT-PROC (BR: class bioreservoir. C: float . ET: integer. win: class window. dlg: class dialog)
  PO = the biopool upon the subworkspace of BR:
  top = the top-container that is the-container-of-br BR:
  define a local float EV:
  if BR is a sol-mol-reservoir or BR is a exp-cell-reservoir then
    EV = C {molar} * 6.023e23 {molecules/liter} * the volume of top {liters}
  t = the current-simulation-time of dlg:
  incr = the current-simulation-time-increment of dlg:
  wait until t >= ET checking every 1 second:
  conclude that the density-entry of PO = EV:
  wait until t >= ET + incr checking every 1 second:
  conclude that the density-entry of PO = 0.0:
  if t > ET + incr then return
```

```
DENSITY-INPUT-PROC (BR: class bioreservoir. D: float . ET: integer. win: class window. dlg: class dialog)
  PO = the biopool upon the SW of BR:
  t = the current-simulation-time of dlg:
  incr = the current-simulation-time-increment of dlg:
  wait until t >= ET checking every 1 second:
  conclude that the density-entry of PO = D:
  wait until t >= ET + incr checking every 1 second:
  conclude that the density-entry of PO = 0.0:
  if t > ET + incr then return:
```

```
RELATIVE-INPUT-PROC (BR: class bioreservoir. F: float  ET: integer.  win: class window. dlg: class  dialog)
  PO = the biopool upon the subworkspace of BR:
  t = the current-simulation-time of dlg:
  incr = the current-simulation-time-increment of dlg:
  wait until t >= ET checking every 1 second:
  conclude that the scaled-entry of PO = F:
  wait until t >= ET + incr checking every 1 second:
  conclude that the scaled-entry of PO = 0.0:
  if t > ET + incr then return.
```

TABLE 232

```
PAUSE-EXPER-CALLBACK (button: class pushbutton. win: class window. panel:
class entry-panel)

 SW = the subworkspace of panel:
 M = the model-definition upon SW:
 call g2-pause-model (M):
 if there exists a text-pushbutton rm upon SW such that (the id of rm = "rm") then enable rm:
 disable button :
```

```
RESUME-EXPER-CALLBACK (button: class pushbutton. win: class window. panel:
class entry-panel)

 SW = the subworkspace of panel:
 M = the model-definition upon SW:
 call g2-resume-model (M):
 conclude that the model-simulation-status of M is running:
 if there exists a text-pushbutton pa upon SW such that (the id of pa = "pa") then enable pa:
 disable button:
```

TABLE 233

```
RESET-EXPER-CALLBACK (button: class pushbutton. win: class
window. panel: class entry-panel)

 SW = the subworkspace of panel:
 M = the model-definition upon SW:
 if the g2-user-mode of win is simulation then
    call RESET-SIMUL-PROC (M. panel):
    call g2-reset-model (M):
    conclude that the current-simulation-time of panel = 0 seconds:
    conclude that the current-simulation-time-increment of panel = 0
       seconds:
 else
    call RESET-NAVIG-PROC (panel):
 if there exists a text-pushbutton st upon SW such that (the id of st = "st") then disable st:
 if there exists a text-pushbutton pa upon SW such that (the id of pa = "pa") then disable pa:
 if there exists a text-pushbutton rm upon SW such that (the id of rm = "rm") then disable rm:
 if there exists a text-pushbutton hi upon SW such that (the id of hi = "hi") then enable hi:
 for md = each uil-message-dialog upon the subworkspace of DIALOG-BIN do
  delete md:
```

VI. CLAIMS:

What is claimed is:

1. A computer system comprising one or more processors coupled to one or more memory components, storage devices, displays, and user input devices, further comprising a program stored in said memory components for enabling the computer generation of virtual iconic models of complex multidimensional systems, wherein said program further comprise:
   - sets of different types of modular domain-specific processes icons, wherein each of said process icons represent one the many processes of different types in said complex system, together with its inputs and outputs
   - sets of modular domain-specific reservoir icons, wherein each of said reservoir icons represents a pool of a single type/state of entity in said complex system, together with its inputs and outputs;
   - a code library comprising means for establishing links between process icons and reservoir icons, wherein said links are established between each input component of a process icon and an output component of each reservoir icon representing an input source for said process; and wherein said links are established between each output component of a process icon and an input component of each reservoir icon representing an output target for said process;
   - a code library comprising the means for the integration of alternating layers of said processes icons linked to said reservoir icons into multidimensional pathways, wherein each of said reservoir icons receives inputs from one or more of said process icons and/or is the source of inputs to one or more of said process icons, and wherein each of said process icons receives inputs from one or more of said reservoir icons and/or is the source of inputs to one or more of said reservoir icons;
2. An system based on simulated virtual iconic models of complex processing systems for enabling the monitoring and control of the operation of said processing systems, wherein said virtual models comprise multidimensional pathways of linked and alternating processes icons and reservoir icons, wherein the process icons represent different types of processes in said processing system, together with their inputs and outputs, wherein each of said reservoir icons represents a pool of a single type/state of entity in said system, together with its inputs and outputs, wherein each of said reservoir icons receives inputs from one or more of said process icons and/or is the source of inputs to one or more of said process icons, wherein each of said process icons receives inputs from one or more of said reservoir icons and/or is the source of inputs to one or more of said reservoir icons, and wherein the system further comprises:
   - a computer system comprising one or more processors coupled to one or more memory units, storage devices, displays, and user input devices;
   - a reactor, wherein the reactor may comprise any combination of reaction tanks, fermentors, bioreactors, or any other domain-specific processing systems, such as ecological habitats, environmental systems, manufacturing systems, or work places;
   - one or more monitoring devices linked to said computer system, wherein said devices monitor the operation of said reactor, and wherein said devices may comprise any combination of hardware, software, or manual data acquisition devices, such as sensors or other devices that measure values related to certain entities in the reactor, signal transducers, filters, amplifiers, data acquisition boards, appropriate device drivers, or any other required devices;
   - one or more controller devices linked to said computer system, wherein said devices regulate the operation of said reactor, such as controlling the flow of certain inputs to the reactor;
   - one or more domain-specific knowledge-bases loaded on said computer system describing virtual models of said complex processing systems, wherein said virtual models encapsulate variables, parameters and values characteristic of said processing systems for enabling the real-time simulation of said processing systems, and wherein one or more of said variables represent values of certain entities in the reactor;
   - a main program loaded on said computer system, wherein the components of said main program comprise:
     - monitoring components enabling the use of values monitored in said reactor to provide inputs to the simulation components and/or the control components;
     - simulation components enabling the real-time simulation of said virtual models based on the information and data contained in said domain-specific knowledge-bases, with or without additional inputs from said monitoring components;
     - control components enabling the generation of control signals that are based on said simulation of the virtual models, wherein said control signals may or may not be additionally based on inputs from said monitoring components, and wherein said control signals are passed to said controller devices for regulating the operation of said reactor.

- one or more monitoring interfaces loaded on said computer system, wherein said interfaces provide software bridges between said monitoring devices and said main program enabling passing of values measured for certain entities in the reactor to the corresponding variables encapsulated in the virtual models that represent said values being measured;
- one or more controller interfaces loaded on said computer system, wherein said interfaces provide software bridges between said main program and said controller devices enabling passing of control signals generated by said main program based on the real-time simulation of said virtual models.

3. A system as described in any of the preceding claims, further comprising location icons representing discrete space compartments, enabling to visually organize the large amounts of components of said virtual models into multiple encapsulated layers of compartments representing the actual location of the entities or processes of the complex systems represented by said components:

4. A system as described in any of the preceding claims, further comprising time-compartment icons representing discrete time compartments, enabling to visually organize the large amounts of components of said virtual models into a series of successive compartments representing the actual time sequence of ordered sets of the processes of the complex systems represented by said components:

5. A system as described in any of the preceding claims, further comprising entity icons representing the description of representative units of the entities of the pools represented by said reservoir icons, wherein said description can be accessed from the reservoir icons.

6. A system as described in any of the preceding claims, wherein each process icon represents a composite object with an associated subworkspace, wherein said subworkspace holds a predefined set of the modular iconic components of each process icon enabling visual access to each of said components, wherein said subworkspace can be displayed or hidden enabling the hiding or visual display of said components, and wherein said components of each process icon further comprise:
   - reactant icons representing the different inputs to said process icon, wherein different types of reactant icons represent the different roles that said inputs play as participants in the process represented by the process icon, wherein each of said reactant icons is to be linked to one of the input components of a reservoir;
   - an engine icon representing the action performed by the process represented by said process icon;
   - product icons representing the different outputs from the process represented by said process icon, wherein each of said product icons is to be linked to one of the input components of a reservoir;

7. A system as described in any of the preceding claims, wherein each reservoir icon represents a composite object with an associated subworkspace, wherein said subworkspace holds a predefined set of the modular iconic components of each reservoir icon enabling visual access to each of said components, wherein said subworkspace can be displayed or hidden enabling the hiding or visual display of said components, and wherein said components of each reservoir icon further comprise:
   - a pool icon representing a pool of a single type/state of entities;
   - a set of input-connector icons connected to said pool icon, wherein each of said connector icons represent each of the inputs to said pool and is to be linked to one of the output components of a process, and
   - a set of output-connector icons connected to said pool icon, wherein each of said connector icons represent each of the outputs to said pool and is to be linked to one of the input components of a process.

8. A system as described in any of the preceding claims, wherein each of the processes icons and reservoir icons of the virtual model encapsulates one or more variables, parameters or values that represent quantities, rates, coefficients or other quantitative or symbolic values, enabling to distribute the components of the mathematical models or inference models that characterize the behavior of the complex system among all its modular components, and further enabling the manipulation of any of those values for any icon independently of the others.

9. A system as described in any of the preceding claims, further comprising libraries of iconic prebuilt building-blocks with associated methods for enabling the visual and interactive building of virtual models of complex systems, wherein said libraries further comprise:
   - a library of different types of modular domain-specific process icons representing different types of domain-specific processes;
   - a library of different types of modular domain-specific reservoir icons representing pools of different types of domain-specific entities; and
   - a code library comprising the means for interactively create and link application-specific sets of process icons and reservoir icons, enabling the rapid creation or expansion of virtual models.

10. A system as described in any of the preceding claims, further comprising a program for enabling the dynamic generation of interactive iconic displays of the interconnected components of the multidimensional pathways of said virtual models, wherein said program further comprise:
    - a graphic user interface with means for enabling the selection or control of said dynamic displays of said pathways; and
    - a code library comprising the means for executing the inference processing required for said dynamic generation of displays of pathways and other related tasks requested by the user from said graphic user interface.
11. A system as described in any of the preceding claims, further comprising a program for enabling the navigation through the components of said virtual models, wherein said program further comprise:
    - a graphic user interface with means for enabling the start or control of said navigation; and
    - a code library comprising the means for executing the inference processing required for said the navigation tasks requested by the user from said graphic user interface.
12. A system as described in any of the preceding claims, further comprising a program for enabling the real-time simulation of said virtual models, wherein said program further comprise:
    - a graphic user interface with means for enabling the start or control of said simulation; and
    - a code library comprising the means for executing the inference and simulation processing required for said simulation and other related tasks requested by the user from said graphic user interface.
13. A system as described in any of the preceding claims, further comprising a program comprising the means to perform mixed-type simulations of said virtual models, combining quantitative, semi-quantitative and heuristic methods, enabling the combination of various types of information and data of different degrees of precision.
14. A system as described in any of the preceding claims, further comprising a program comprising the means to perform simulations of said virtual models, , wherein said program comprise methods capable of reasoning during said simulations about values or events over time.
15. A system as described in any of the preceding claims, further comprising a program for enabling the performance of queries based on the different types of components of said virtual models, wherein said program further comprise:
    - a graphic user interface with means for enabling the selection of the constraints or the display the results of of said queries; and
    - a code library comprising the means for executing the complex inference processing required for said queries and other related tasks requested by the user from said graphic user interface.
16. A system as described in any of the preceding claims, wherein the virtual models of complex systems represent chemical systems.
17. A system as described in any of the preceding claims, wherein the virtual models of complex systems represent biological cellular systems.

FIG.1

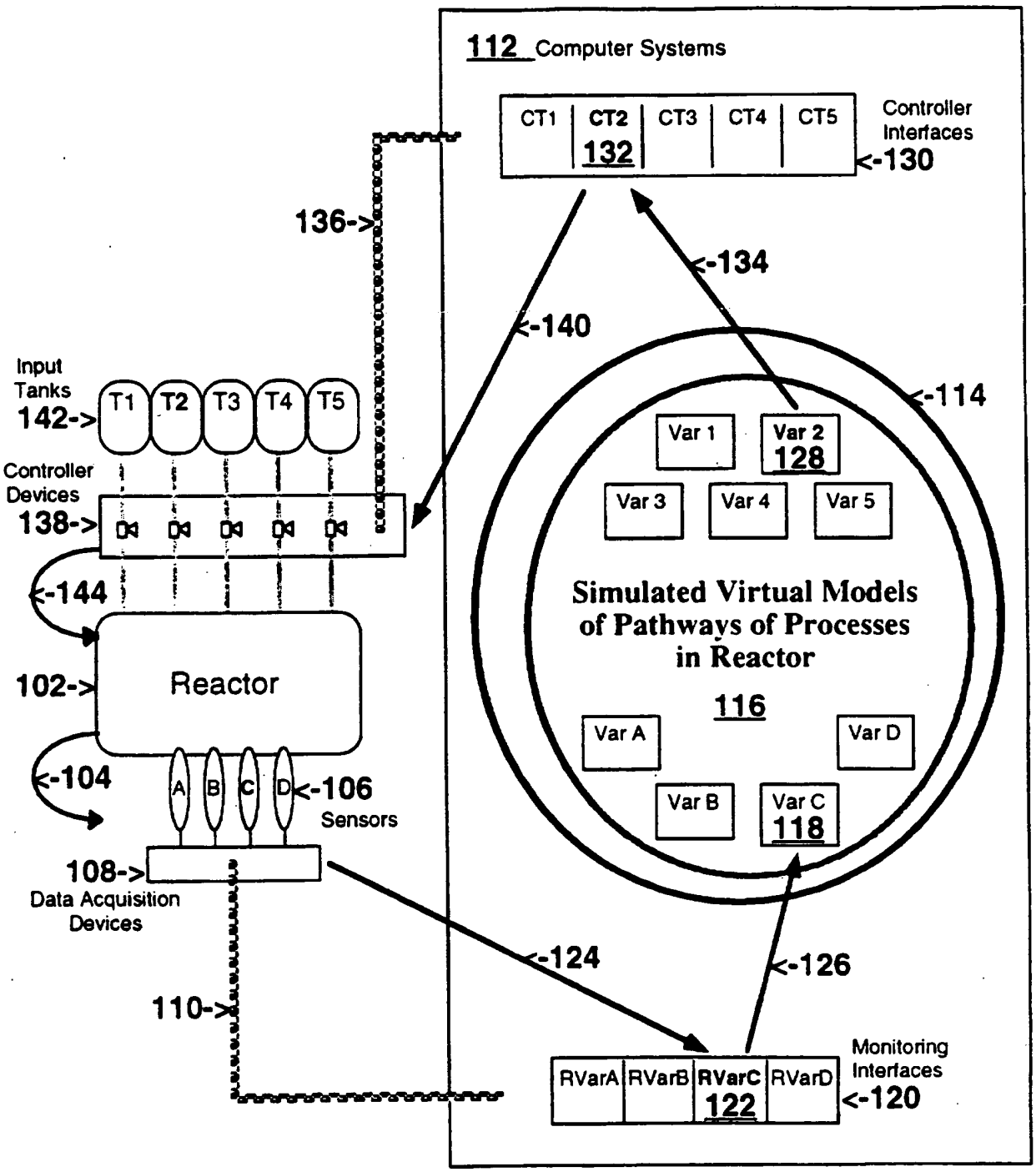

112 Computer Systems
CT1
CT2
132
CT3
CT4
CT5
Controller Interfaces
-130
136->
-134
-140
-114
Input Tanks
142->
T1
T2
T3
T4
T5
Controller Devices
138->
Var 1
Var 2
128
Var 3
Var 4
Var 5
Simulated Virtual Models of Pathways of Processes in Reactor
116
-144
102->
Reactor
Var A
Var D
Var B
Var C
118
-104
A
B
C
D
-106
Sensors
108->
Data Acquisition Devices
-124
-126
110->
RVarA
RVarB
RVarC
122
RVarD
Monitoring Interfaces
-120

FIG.2

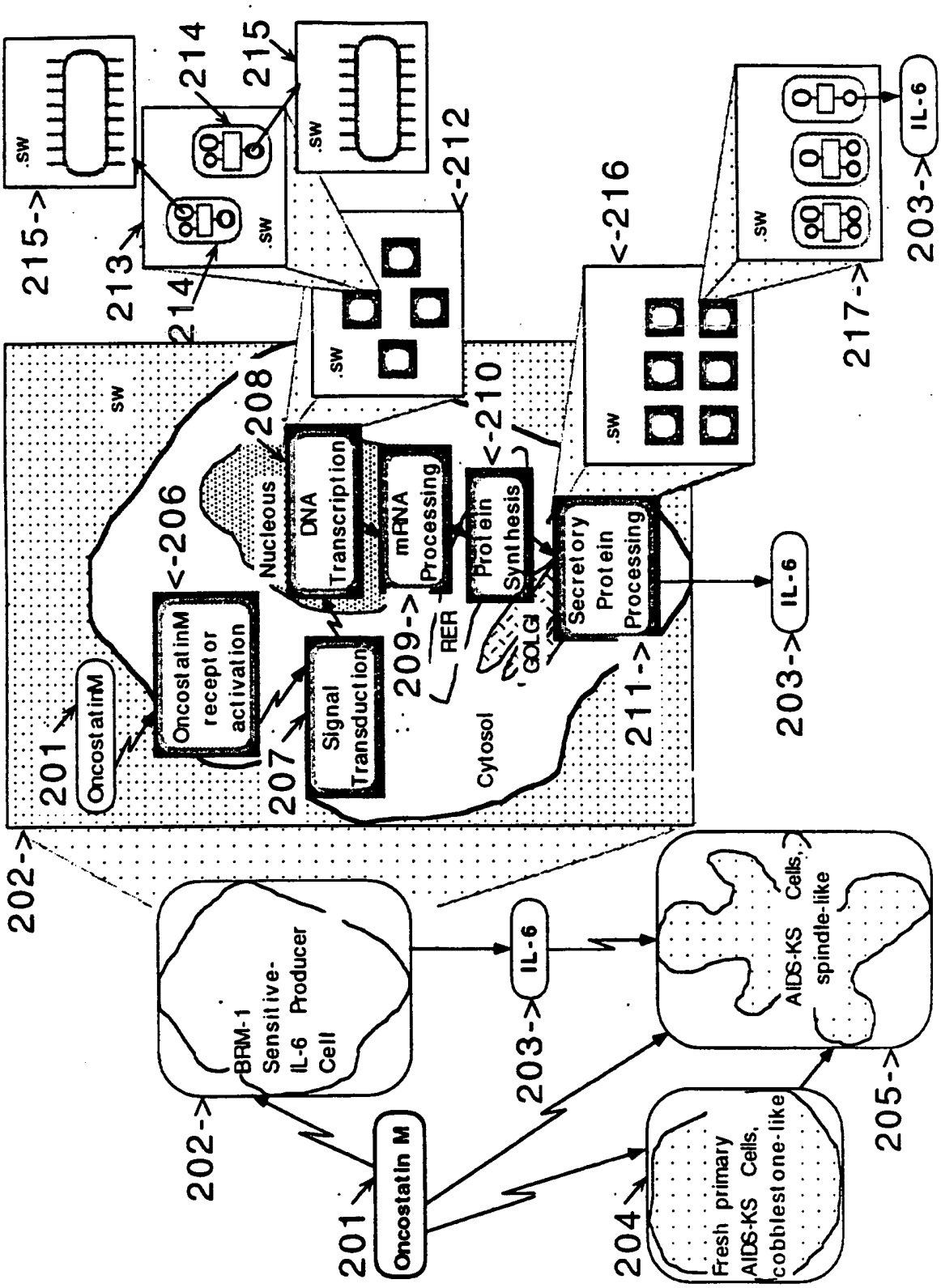

202->
201
Oncostatin M
BRM-1 Sensitive-IL-6 Producer Cell
IL-6
203->
Oncostatin M
receptor activation
<-206
Signal Transduction
207
Nucleous
DNA Transcription
208
mRNA Processing
209->
RER
Protein Synthesis
<-210
GOLGI
Secretory Protein Processing
211->
Cytosol
IL-6
203->
<-212
213
214
215
215->
<-216
217->
IL-6
203->
AIDS-KS Cells, spindle-like
205->
204
Fresh primary AIDS-KS Cells, cobblestone-like
SW

FIGURE 3

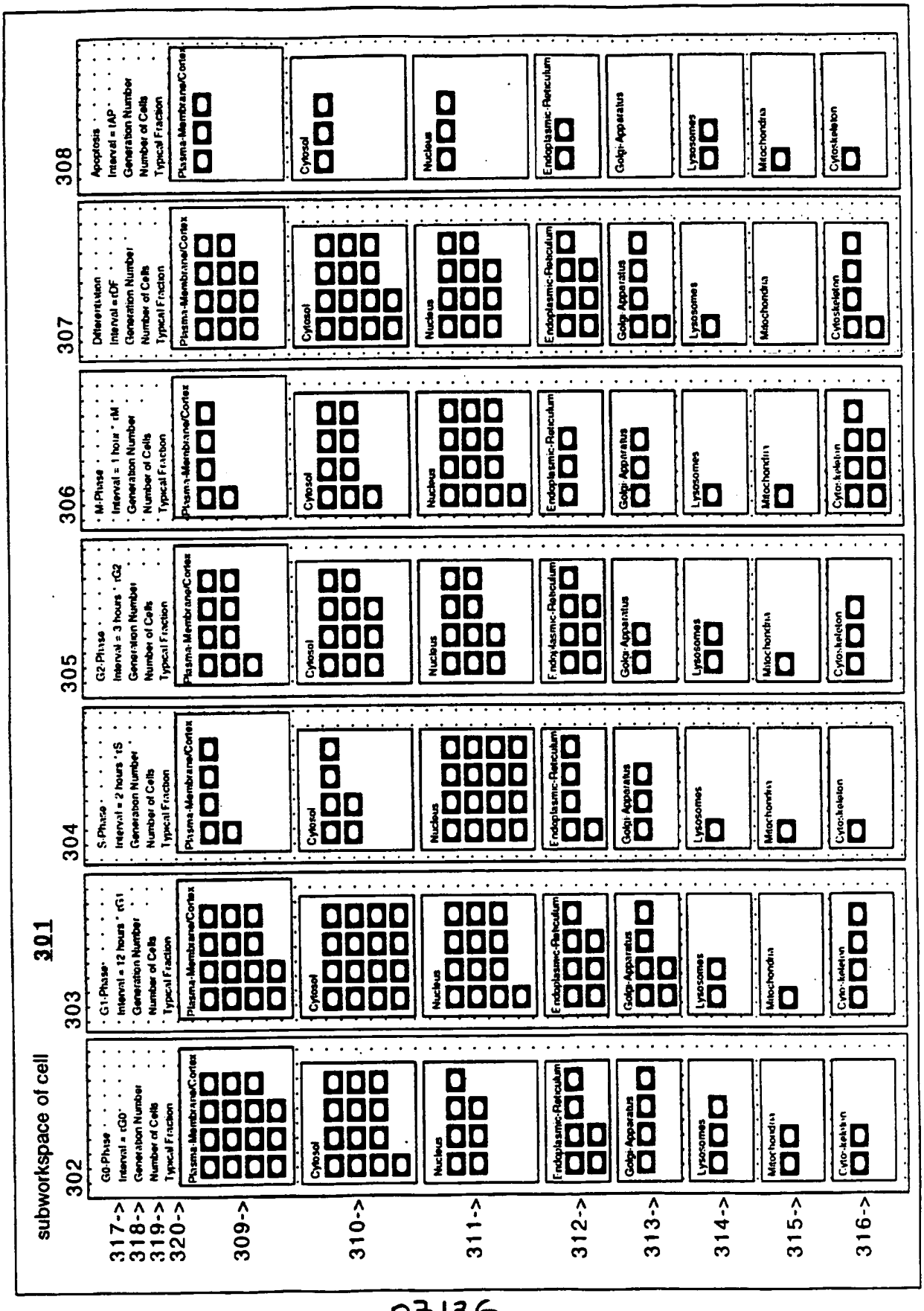

subworkspace of cell
301
302
303
304
305
306
307
308
G0-Phase
Interval = rG0
G1-Phase
Interval = 12 hours * rG1
S-Phase
Interval = 2 hours * rS
G2-Phase
Interval = 3 hours * rG2
M-Phase
Interval = 1 hour * rM
Differentiation
Interval = rDF
Apoptosis
Interval = rAP
Generation Number
Number of Cells
Typical Fraction
Plasma-Membrane/Cortex
Cytosol
Nucleus
Endoplasmic-Reticulum
Golgi-Apparatus
Lysosomes
Mitochondria
Cytoskeleton
309->
310->
311->
312->
313->
314->
315->
316->
317->
318->
319->
320->
03136

FIG.4

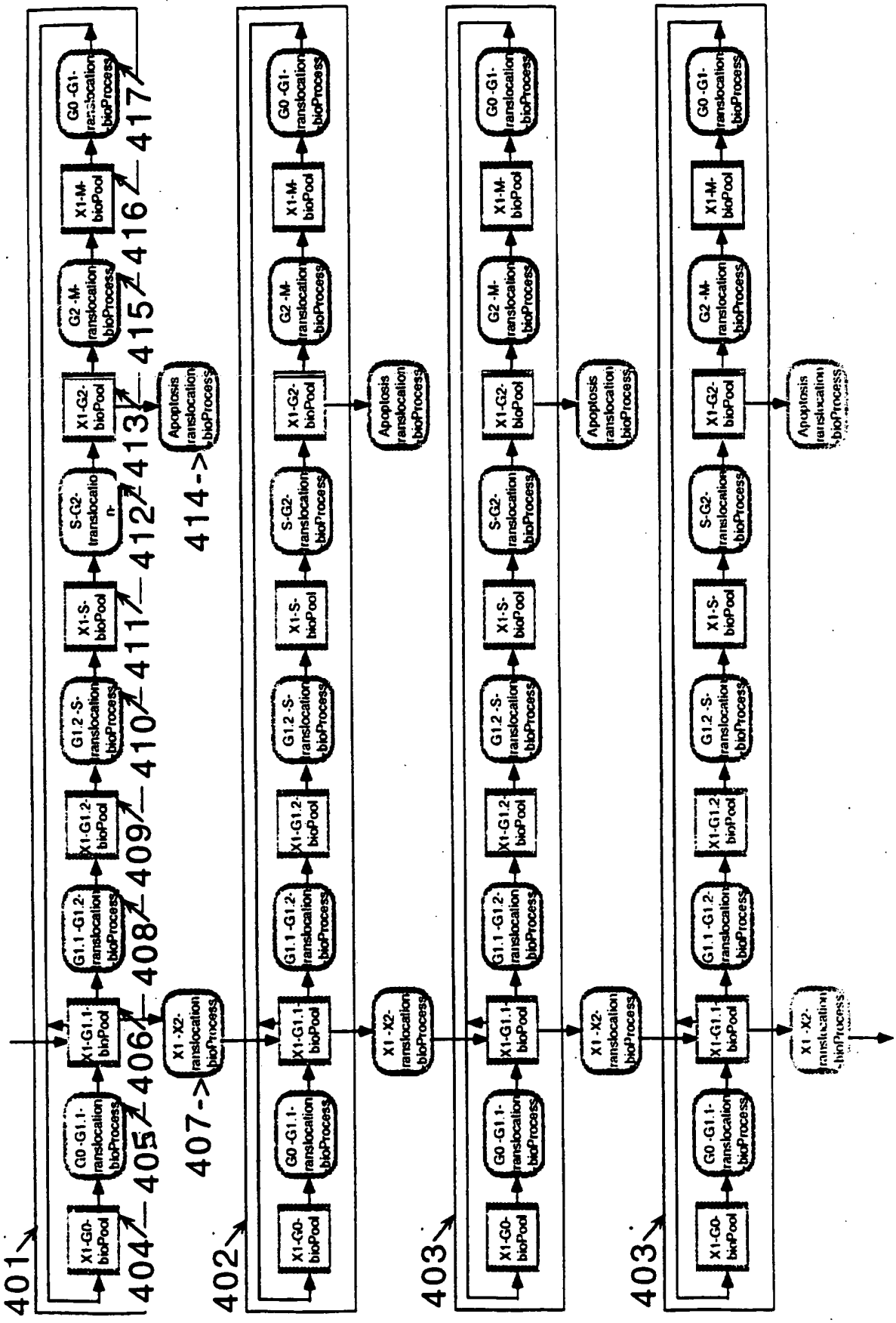

401
402
403
403
404
405
406
407->
408
409
410
411
412
413
414->
415
416
417
X1-G0-bioPool
G0-G1.1-translocation bioProcess
X1-G1.1-bioPool
X1-X2-translocation bioProcess
G1.1-G1.2-translocation bioProcess
X1-G1.2-bioPool
G1.2-S-translocation bioProcess
X1-S-bioPool
S-G2-translocation
X1-G2-bioPool
Apoptosis translocation bioProcess
G2-M-translocation bioProcess
X1-M-bioPool
G0-G1-translocation bioProcess

FIG.5

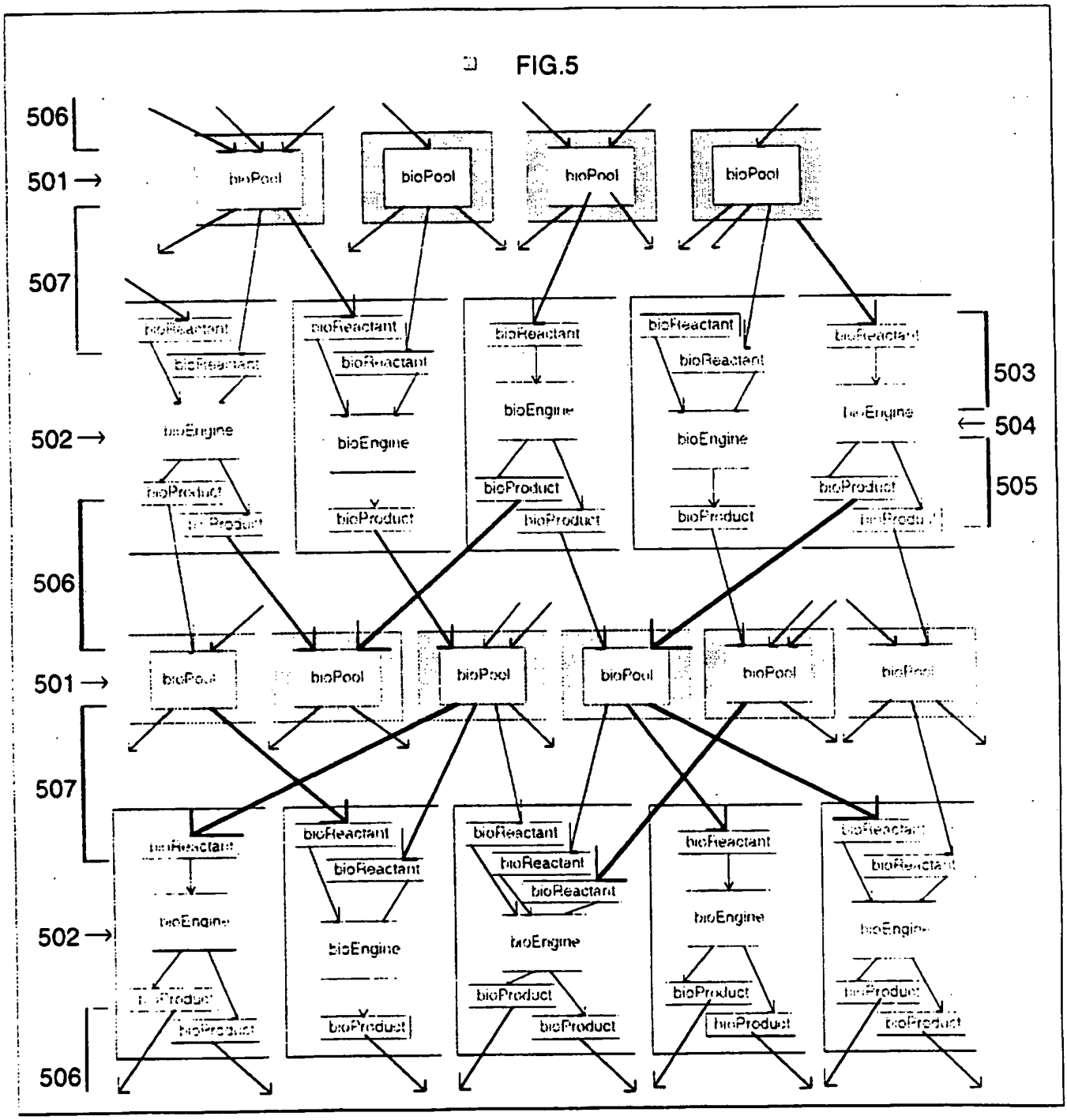

506
501
507
502
503
504
505
bioPool
bioReactant
bioEngine
bioProduct

FIG.6

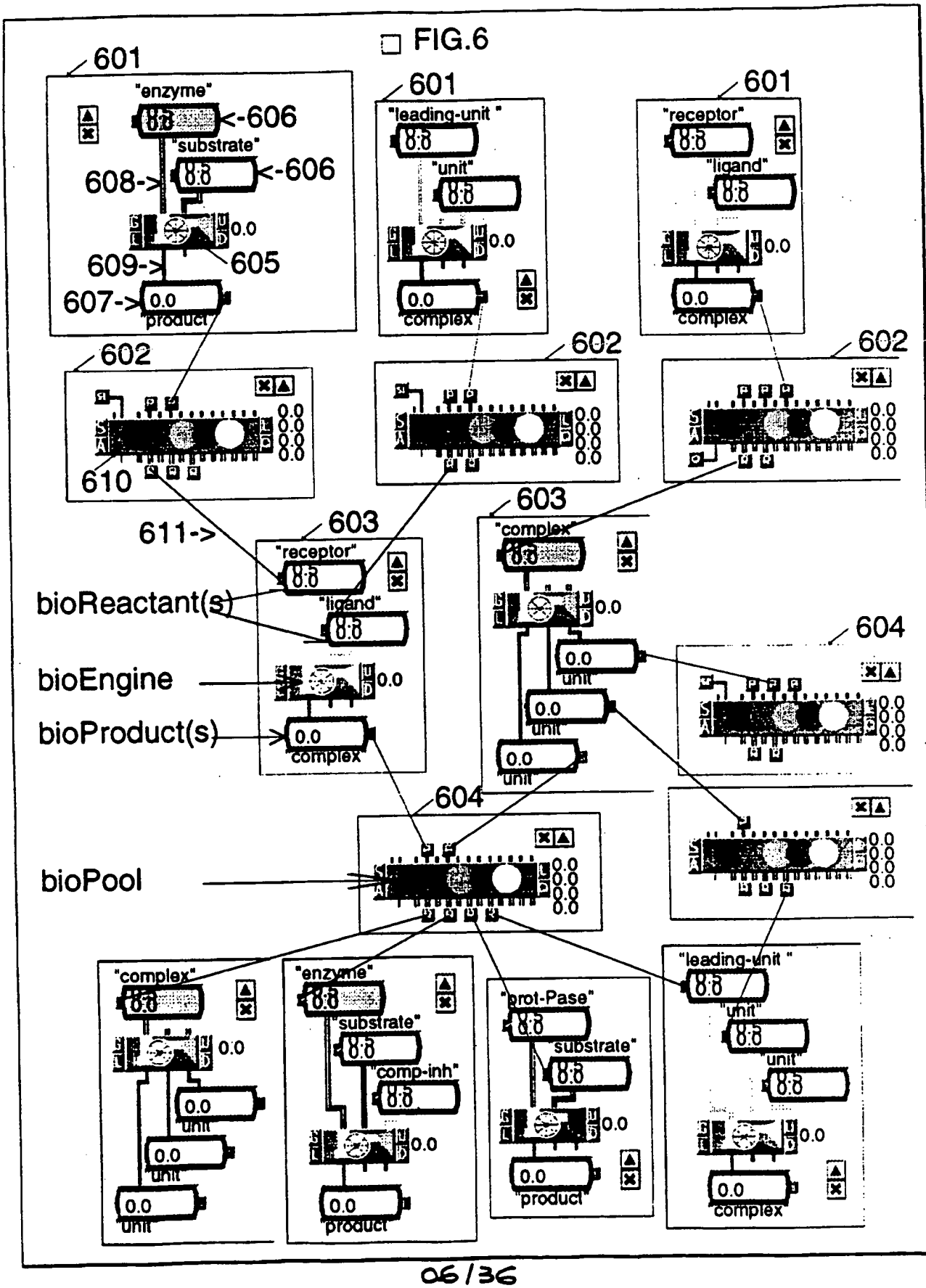

601
"enzyme"
-606
"substrate"
608->
-606
0.0
609->
605
607->
0.0
"product"
601
"leading-unit "
"unit"
0.0
0.0
"complex"
601
"receptor"
"ligand"
0.0
0.0
"complex"
602
610
602
602
611->
603
"receptor"
"ligand"
bioReactant(s)
bioEngine
0.0
bioProduct(s)
0.0
"complex"
603
"complex"
0.0
0.0
"unit"
0.0
"unit"
604
604
bioPool
"complex"
0.0
0.0
"unit"
0.0
"unit"
0.0
"unit"
"enzyme"
"substrate"
"comp-inh"
0.0
0.0
"product"
"prot-Pase"
"substrate"
0.0
0.0
"product"
"leading-unit "
"unit"
"unit"
0.0
0.0
"complex"

FIG.7

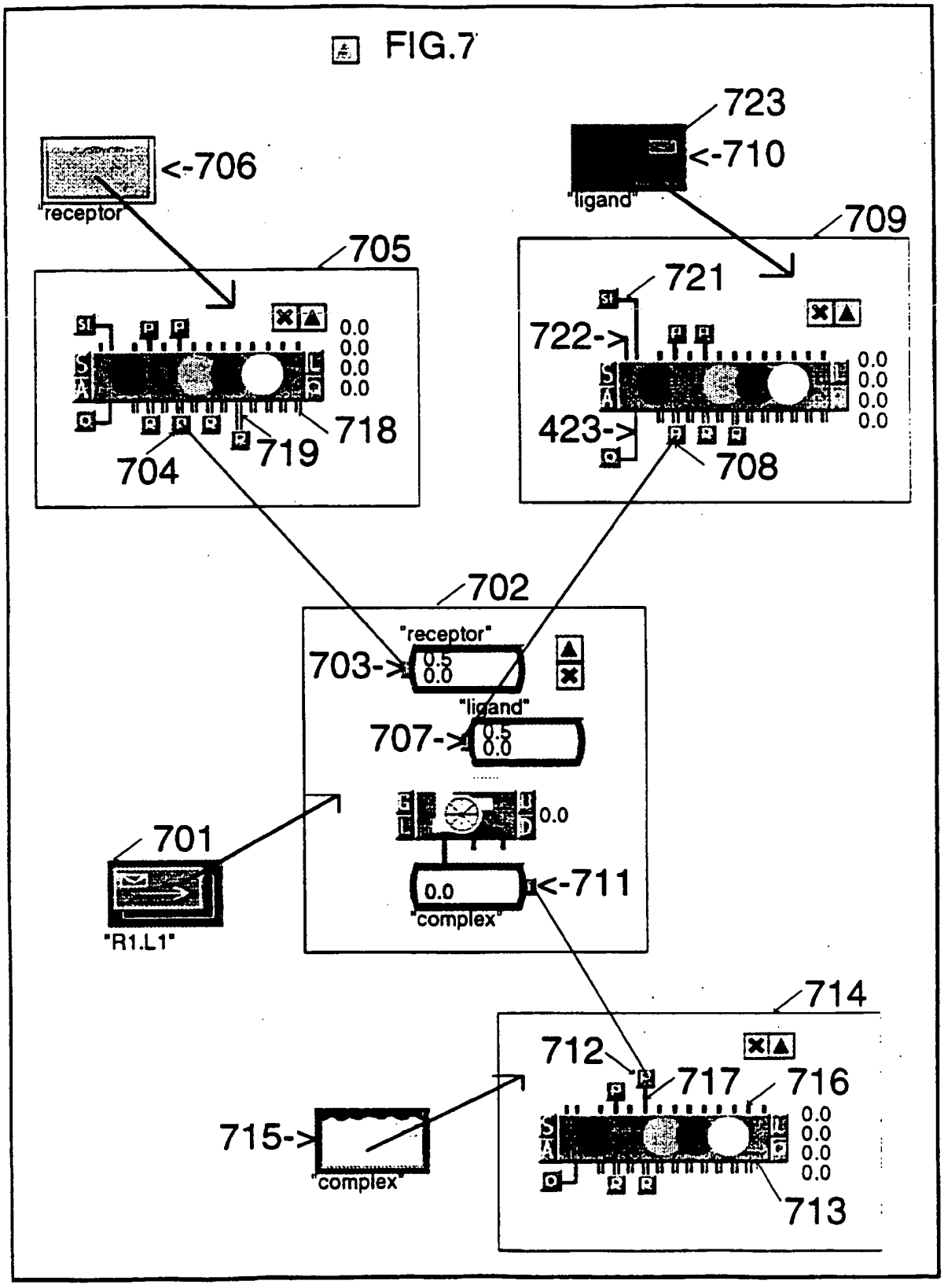

723
<-710
"ligand"
<-706
"receptor"
709
705
721
722->
0.0
0.0
0.0
0.0
718
719
704
423->
708
702
"receptor"
703->
0.5
0.0
"ligand"
707->
0.5
0.0
0.0
701
<-711
0.0
"complex"
"R1.L1"
714
712
717
716
715->
"complex"
0.0
0.0
0.0
0.0
713

FIG.8

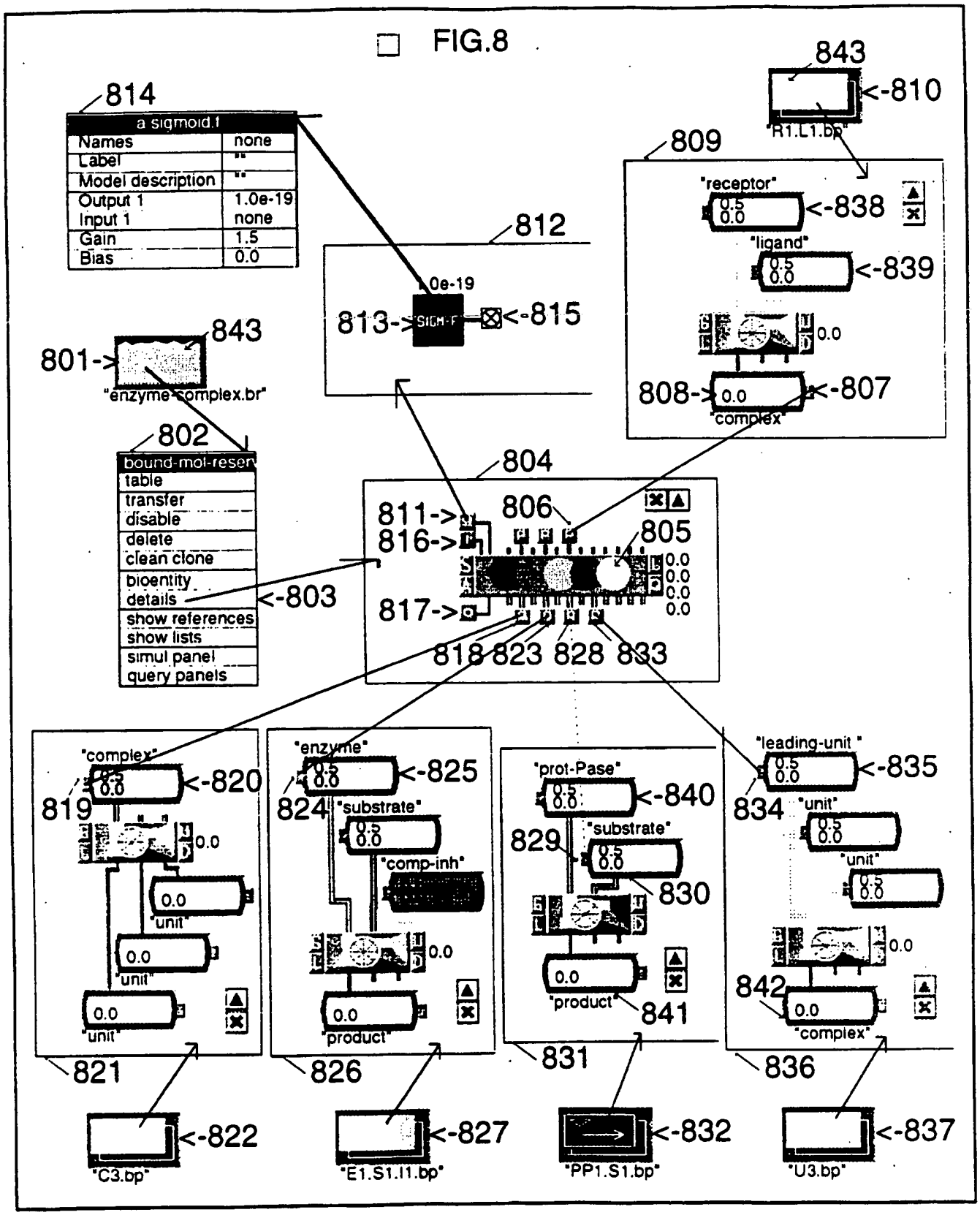

814
a sigmoid.1
Names none
Label ""
Model description ""
Output 1 1.0e-19
Input 1 none
Gain 1.5
Bias 0.0
812
0e-19
813->
SIGM-F
<-815
843
801->
"enzyme-complex.br"
802
bound-mol-reser
table
transfer
disable
delete
clean clone
bioentity details
<-803
show references
show lists
simul panel
query panels
843
<-810
"R1.L1.bp"
809
"receptor"
0.5
0.0
<-838
"ligand"
<-839
0.0
808->
<-807
"complex"
804
811->
816->
806
805
817->
818 823 828 833
"complex"
819
<-820
"unit"
"unit"
"unit"
821
"enzyme"
824
<-825
"substrate"
"comp-inh"
"product"
826
"prot-Pase"
<-840
829
"substrate"
830
"product"
841
831
"leading-unit "
834
<-835
"unit"
"unit"
842
"complex"
836
<-822
"C3.bp"
<-827
"E1.S1.I1.bp"
<-832
"PP1.S1.bp"
<-837
"U3.bp"

FIG.10

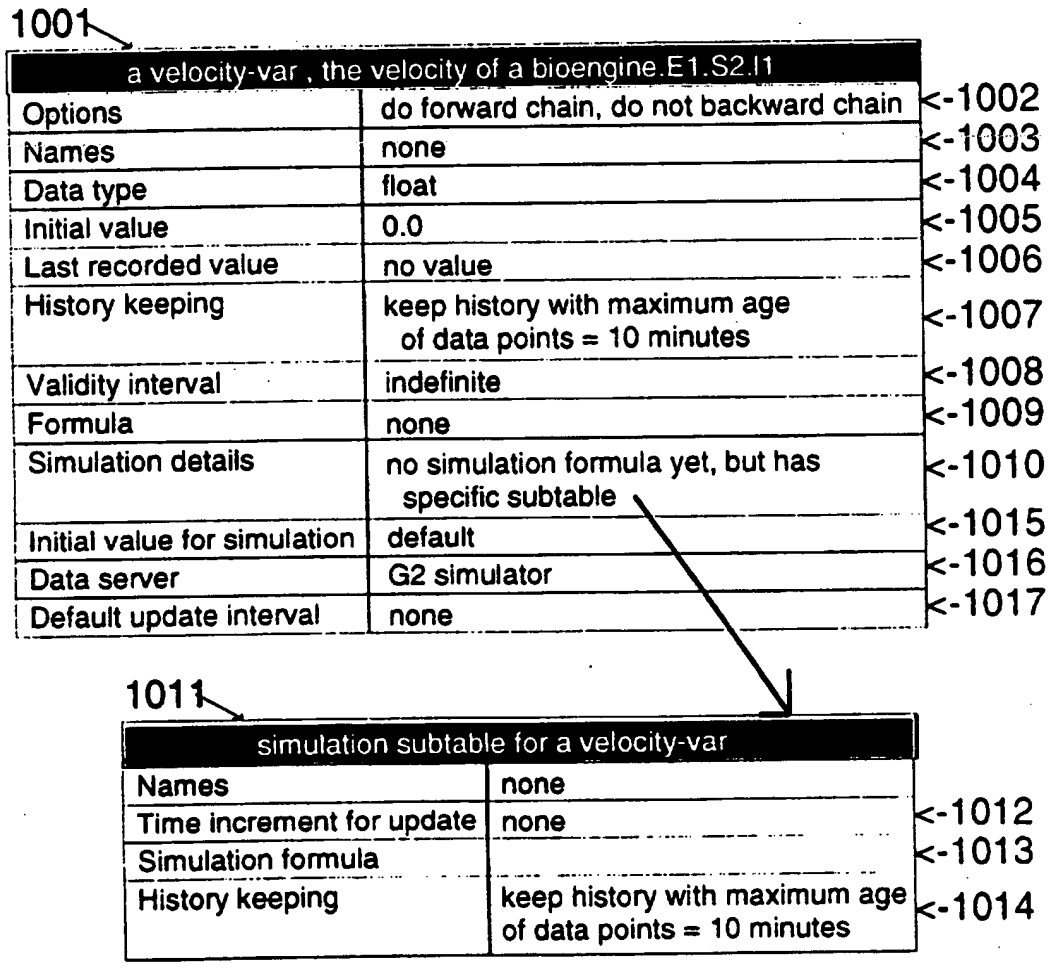

1001
a velocity-var , the velocity of a bioengine.E1.S2.I1
Options
do forward chain, do not backward chain
<-1002
Names
none
<-1003
Data type
float
<-1004
Initial value
0.0
<-1005
Last recorded value
no value
<-1006
History keeping
keep history with maximum age of data points = 10 minutes
<-1007
Validity interval
indefinite
<-1008
Formula
none
<-1009
Simulation details
no simulation formula yet, but has specific subtable
<-1010
Initial value for simulation
default
<-1015
Data server
G2 simulator
<-1016
Default update interval
none
<-1017
1011
simulation subtable for a velocity-var
Names
none
Time increment for update
none
<-1012
Simulation formula
<-1013
History keeping
keep history with maximum age of data points = 10 minutes
<-1014

FIG.11

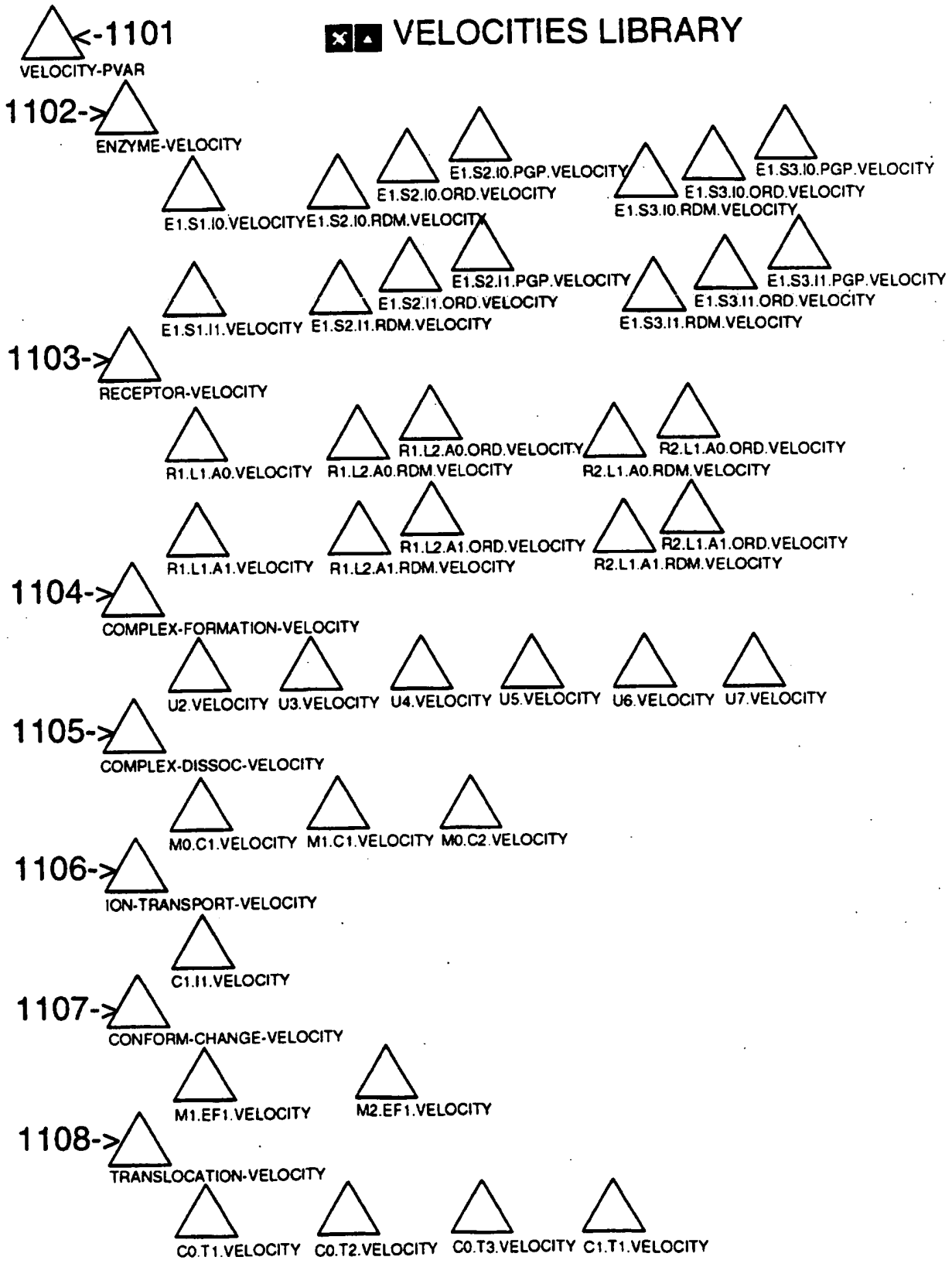

VELOCITIES LIBRARY
1101
VELOCITY-PVAR
1102
ENZYME-VELOCITY
E1.S1.I0.VELOCITY
E1.S2.I0.RDM.VELOCITY
E1.S2.I0.ORD.VELOCITY
E1.S2.I0.PGP.VELOCITY
E1.S3.I0.RDM.VELOCITY
E1.S3.I0.ORD.VELOCITY
E1.S3.I0.PGP.VELOCITY
E1.S1.I1.VELOCITY
E1.S2.I1.RDM.VELOCITY
E1.S2.I1.ORD.VELOCITY
E1.S2.I1.PGP.VELOCITY
E1.S3.I1.RDM.VELOCITY
E1.S3.I1.ORD.VELOCITY
E1.S3.I1.PGP.VELOCITY
1103
RECEPTOR-VELOCITY
R1.L1.A0.VELOCITY
R1.L2.A0.RDM.VELOCITY
R1.L2.A0.ORD.VELOCITY
R2.L1.A0.RDM.VELOCITY
R2.L1.A0.ORD.VELOCITY
R1.L1.A1.VELOCITY
R1.L2.A1.RDM.VELOCITY
R1.L2.A1.ORD.VELOCITY
R2.L1.A1.RDM.VELOCITY
R2.L1.A1.ORD.VELOCITY
1104
COMPLEX-FORMATION-VELOCITY
U2.VELOCITY
U3.VELOCITY
U4.VELOCITY
U5.VELOCITY
U6.VELOCITY
U7.VELOCITY
1105
COMPLEX-DISSOC-VELOCITY
M0.C1.VELOCITY
M1.C1.VELOCITY
M0.C2.VELOCITY
1106
ION-TRANSPORT-VELOCITY
C1.I1.VELOCITY
1107
CONFORM-CHANGE-VELOCITY
M1.EF1.VELOCITY
M2.EF1.VELOCITY
1108
TRANSLOCATION-VELOCITY
C0.T1.VELOCITY
C0.T2.VELOCITY
C0.T3.VELOCITY
C1.T1.VELOCITY

FIG.12

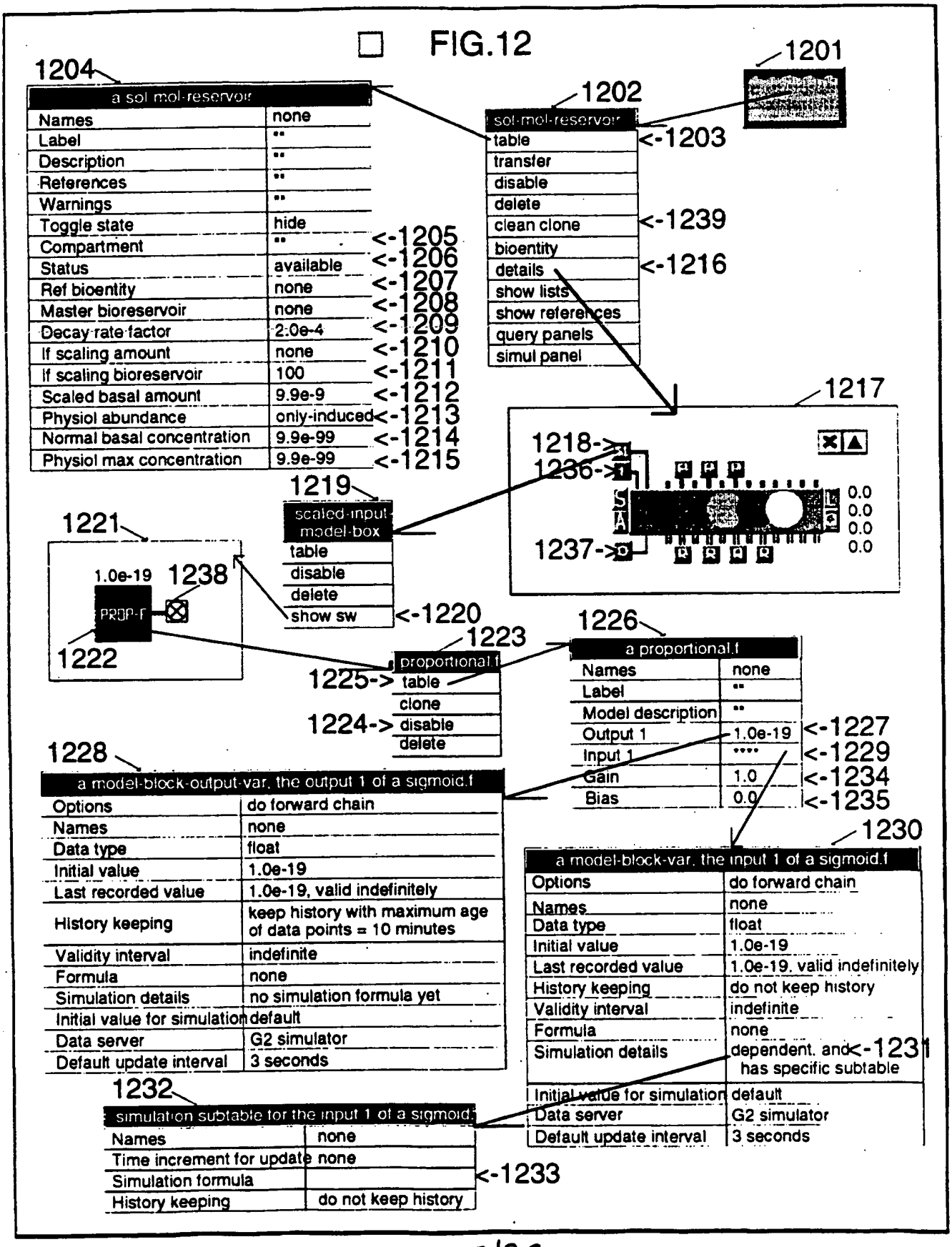
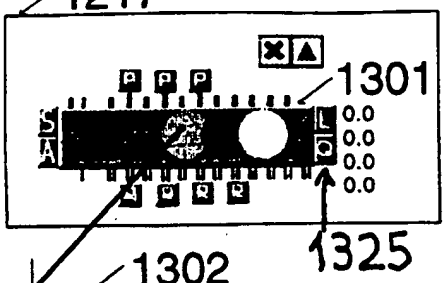

1204
a sol mol-reservoir
Names none
Label ""
Description ""
References ""
Warnings ""
Toggle state hide
Compartment "" <-1205
Status available <-1206
Ref bioentity none <-1207
Master bioreservoir none <-1208
Decay rate factor 2.0e-4 <-1209
If scaling amount none <-1210
If scaling bioreservoir 100 <-1211
Scaled basal amount 9.9e-9 <-1212
Physiol abundance only-induced <-1213
Normal basal concentration 9.9e-99 <-1214
Physiol max concentration 9.9e-99 <-1215
1202
sol-mol-reservoir
table <-1203
transfer
disable
delete
clean clone <-1239
bioentity
details <-1216
show lists
show references
query panels
simul panel
1201
1217
1218->
1236->
1237->
0.0
0.0
0.0
0.0
1219
scaled-input model-box
table
disable
delete
show sw <-1220
1221
1.0e-19
1238
PROP-F
1222
1223
proportional.f
1225-> table
clone
1224-> disable
delete
1226
a proportional.f
Names none
Label ""
Model description ""
Output 1 1.0e-19 <-1227
Input 1 **** <-1229
Gain 1.0 <-1234
Bias 0.0 <-1235
1228
a model-block-output-var. the output 1 of a sigmoid.f
Options do forward chain
Names none
Data type float
Initial value 1.0e-19
Last recorded value 1.0e-19, valid indefinitely
History keeping keep history with maximum age of data points = 10 minutes
Validity interval indefinite
Formula none
Simulation details no simulation formula yet
Initial value for simulation default
Data server G2 simulator
Default update interval 3 seconds
1230
a model-block-var. the input 1 of a sigmoid.f
Options do forward chain
Names none
Data type float
Initial value 1.0e-19
Last recorded value 1.0e-19. valid indefinitely
History keeping do not keep history
Validity interval indefinite
Formula none
Simulation details dependent. and has specific subtable <-1231
Initial value for simulation default
Data server G2 simulator
Default update interval 3 seconds
1232
simulation subtable for the input 1 of a sigmoid
Names none
Time increment for update none
Simulation formula <-1233
History keeping do not keep history

FIG.13



1302

sol-mol-pool
table
color
disable
delete
add top post
add bottom post
add scaled input
add input
add output

<-1303

1304

a sol-mol-pool	
Names	none
Label	""
Description	""
Basal density	9.9e-99 UPV
Scaling density	99.999 UPV
Density entry	0.0 UPV
Scaled entry	0.0
Input rate	0.0
Output rate	0.0
Accumulation rate	0.0
Density	9.9e-99 UPV
Scaled amount	0.0
Basal concentration	9.9e-99 M
Concentration	9.9e-99 M

1305 1307 1309 1311 1313 1315 1317 1319 1321 1323

1306

a basal-density-par, the basal density of a sol-mol-pool	
Options	do forward chain
Names	none
Data type	UPV
Initial value	9.9e-99
Last recorded value	9.9e-99 UPV
History keeping	do not keep history

1308

a scaling-density-par, the scaling density o a sol-mol-pool	
Options	do forward chain
Names	none
Data type	UPV
Initial value	99.9e-99
Last recorded value	99.9e-99 UPV
History keeping	do not keep history
Set by user	false

1310

a density-entry-pvar, the density entry of a sol-mol-p	
Options	do forward chain
Names	none
Data type	float
Initial value	0.0
Last recorded value	0.0 UPV
History keeping	keep history with maximum number of data points = 4

1318

a density-pvar, the density of a sol-mol-pool	
Options	do forward chain
Names	none
Data type	UPV
Initial value	9.9e-99
Last recorded value	9.9e-99 UPV
History keeping	keep history with maximum age of data points = 10 minutes

1312

a scaled-entry-pvar, the scaled entry of a sol-mol-p	
Options	do forward chain
Names	none
Data type	float
Initial value	0.0
Last recorded value	0.0
History keeping	keep history with maximum number of data points = 4

1322

a basal-conc-par, the basal concentration o a sol-mol-pool	
Options	do forward chain
Names	none
Data type	M
Initial value	9.9e-99
Last recorded value	9.9e-99 M
History keeping	do not keep history

1314

an input-rate-pvar, the input rate of a sol-mol-pool	
Options	do forward chain
Names	none
Data type	float
Initial value	0.0
Last recorded value	0.0
History keeping	keep history with maximum age of data points = 10 minutes

1324

a conc-pvar, the concentration of a sol-mol-pool	
Options	do forward chain
Names	none
Data type	M
Initial value	9.9e-99
Last recorded value	9.9e-99 M
History keeping	keep history with maximum age of data points = 10 minutes

FIGURE 14

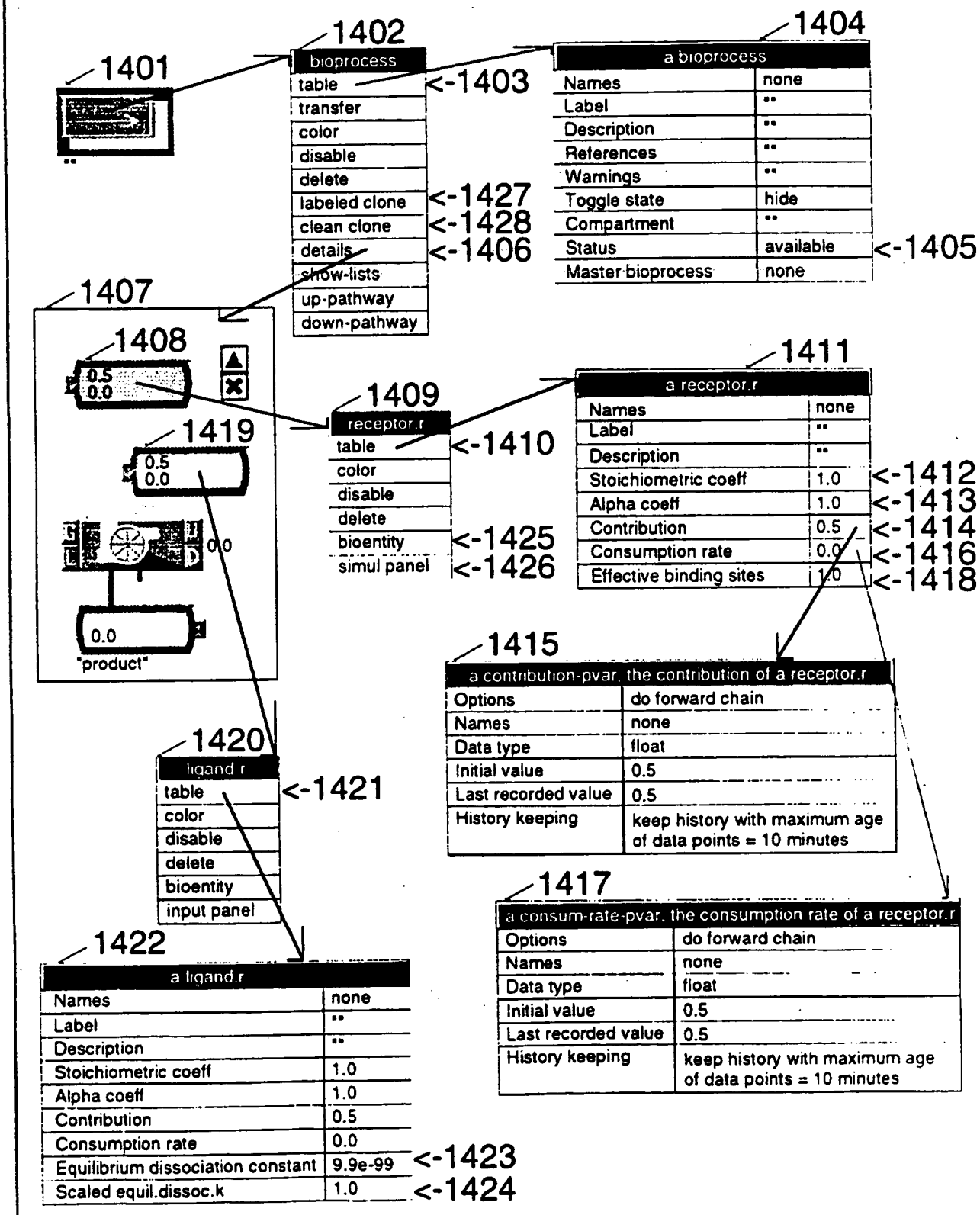
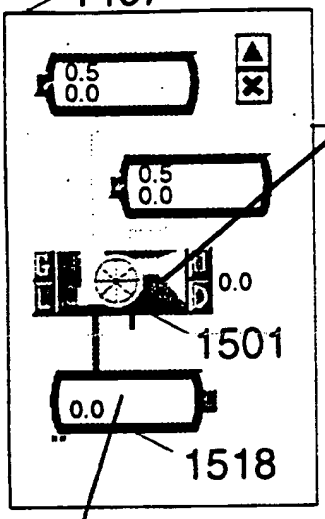

1401
1402
bioprocess
table
transfer
color
disable
delete
labeled clone
clean clone
details
show-lists
up-pathway
down-pathway
<-1403
<-1427
<-1428
<-1406
1404
a bioprocess
Names none
Label ""
Description ""
References ""
Warnings ""
Toggle state hide
Compartment ""
Status available
Master bioprocess none
<-1405
1407
1408
0.5
0.0
1409
receptor.r
table
color
disable
delete
bioentity
simul panel
<-1410
<-1425
<-1426
1411
a receptor.r
Names none
Label ""
Description ""
Stoichiometric coeff 1.0
Alpha coeff 1.0
Contribution 0.5
Consumption rate 0.0
Effective binding sites 1.0
<-1412
<-1413
<-1414
<-1416
<-1418
1419
0.5
0.0
0.0
"product"
1415
a contribution-pvar, the contribution of a receptor.r
Options do forward chain
Names none
Data type float
Initial value 0.5
Last recorded value 0.5
History keeping keep history with maximum age of data points = 10 minutes
1420
ligand r
table
color
disable
delete
bioentity
input panel
<-1421
1417
a consum-rate-pvar, the consumption rate of a receptor.r
Options do forward chain
Names none
Data type float
Initial value 0.5
Last recorded value 0.5
History keeping keep history with maximum age of data points = 10 minutes
1422
a ligand.r
Names none
Label ""
Description ""
Stoichiometric coeff 1.0
Alpha coeff 1.0
Contribution 0.5
Consumption rate 0.0
Equilibrium dissociation constant 9.9e-99
Scaled equil.dissoc.k 1.0
<-1423
<-1424

FIG.15



1502

bioengine.r1
table
color
disable
delete
create sw

<-1503

1504

a bioengine.r1.l1	
Names	none
Label	""
Description	""
Velocity	0.0 <-1505
Rate constant sec	1.0 <-1507
Tau coeff	1.0 <-1508
Bias	0.0 <-1509
Time lag	0 seconds <-1510
pH	7.4 <-1512
pH deviation factor	1.0 <-1514
Temperature	37.0 DEGREES <-1516
Temp deviation factor	1.0 <-1517
Toggle state	hide
Type color	none

1506

a binding-rate-pvar, the velocity of a bioengine.r1.l	
Options	do forward chain
Names	none
Data type	float
Initial value	0.0
Last recorded value	0.0
History keeping	keep history with maximum age of data points = 10 minutes

1519

bioproduct
table <-1520
color
disable
delete
bioentity <-1525

1511

a time-delay-par, the time lag of a bioengine.r	
	do forward chain
	none
Data type	interval
Initial value	0
Last recorded value	0 seconds
History keeping	do not keep history

1521

a bioproduct	
Names	none
Label	""
Description	""
Stoichiometric coeff	1.0
Production rate	0.0 <-1522

1513

a ph-par, the ph of a bioengine.r1.l1	
Options	do forward chain
Names	none
Data type	float
Initial value	7.4
Last recorded value	7.4
History keeping	do not keep history

1523

a produc-rate-pvar, the production rate of a bioproduc	
Options	do forward chain
Names	none
Data type	float
Initial value	0.0
Last recorded value	0.0
History keeping	keep history with maximum age of data points = 10 minutes

1515

a ph-factor-par, the ph deviation factor of a bioengine.r1.l1	
Options	do forward chain
Names	none
Data type	float
Initial value	1.0
Last recorded value	1.0
History keeping	do not keep history

FIG.16

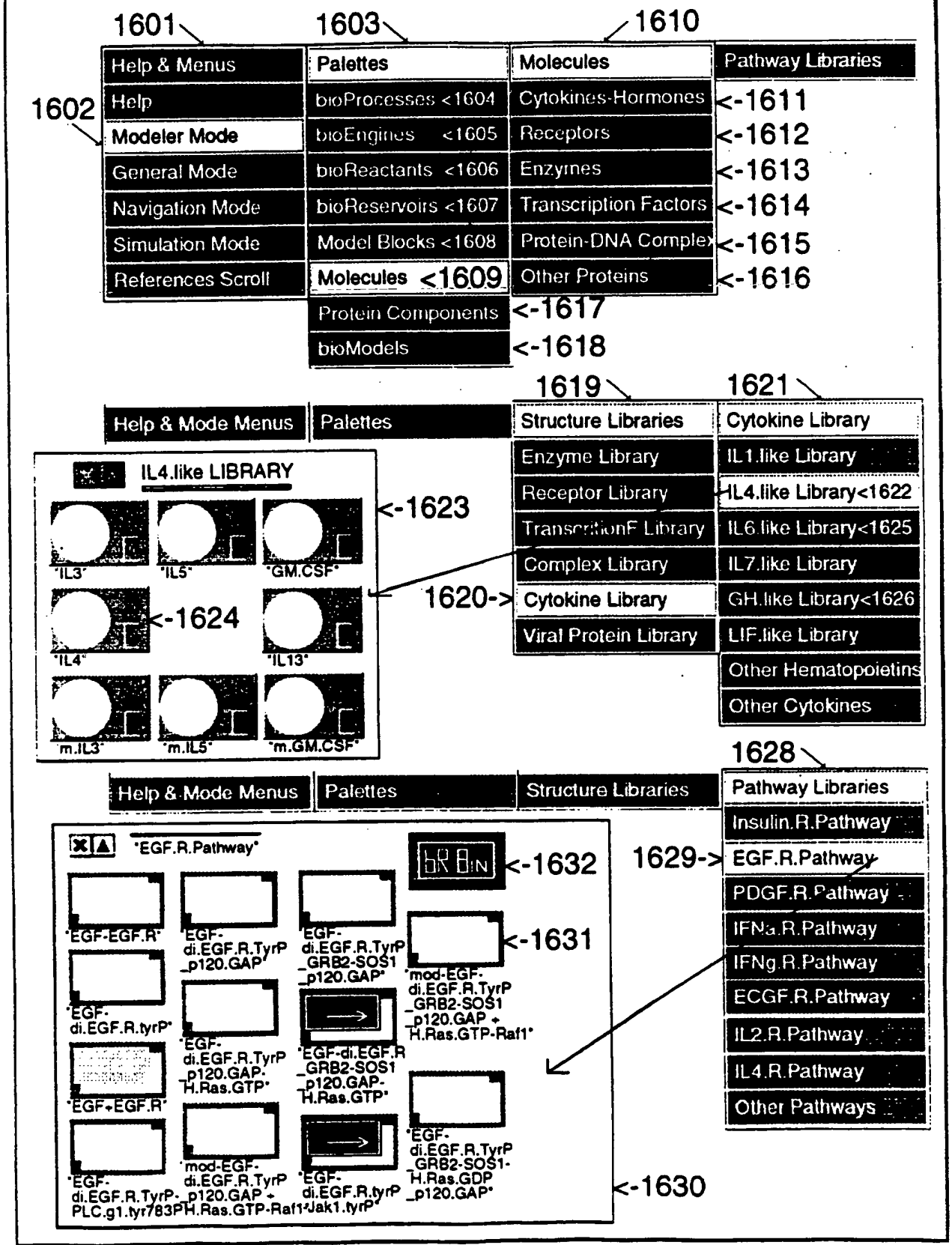
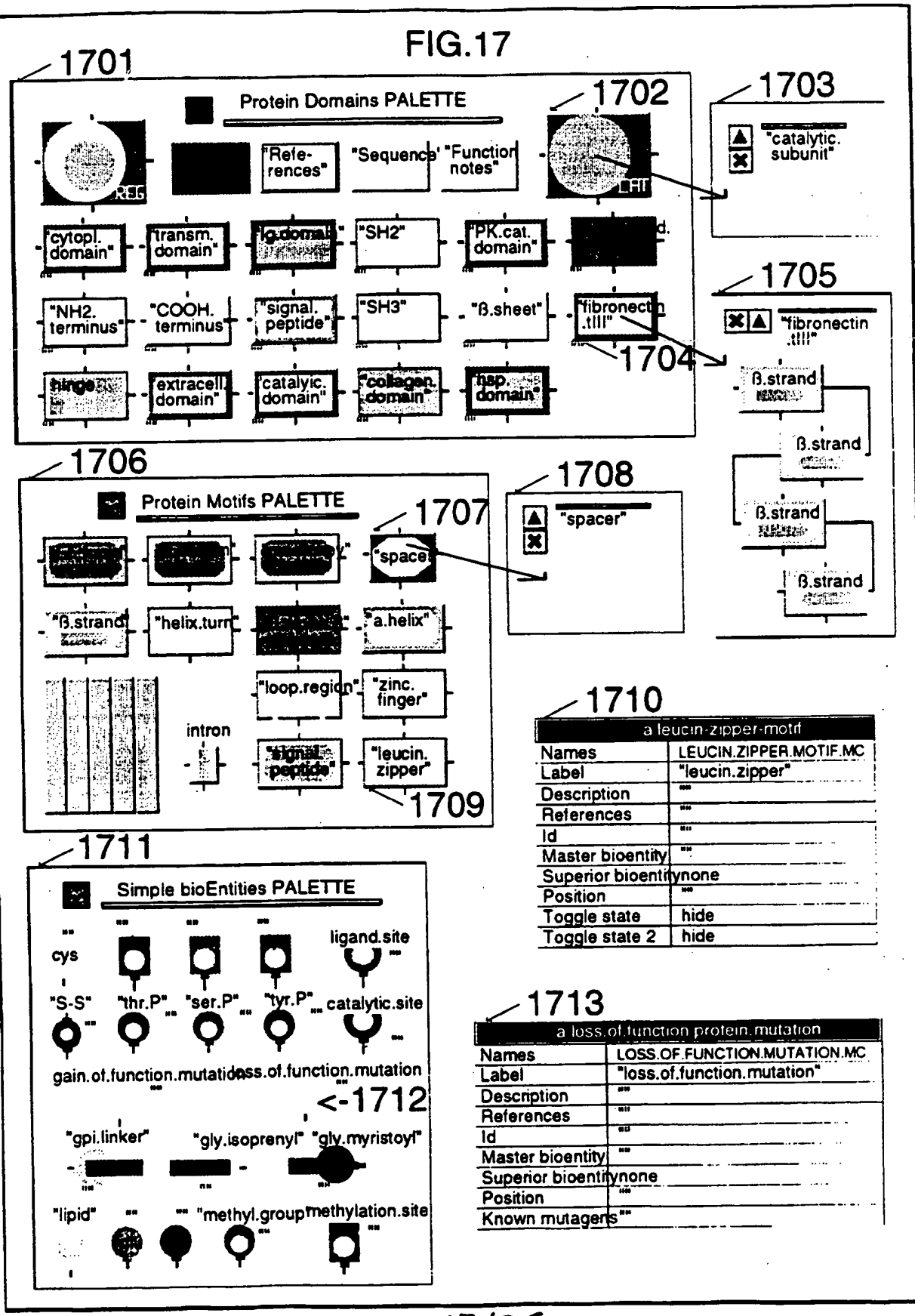

1601
1603
1610
Help & Menus
Palettes
Molecules
Pathway Libraries
1602
Help
Modeler Mode
General Mode
Navigation Mode
Simulation Mode
References Scroll
bioProcesses <1604
bioEngines <1605
bioReactants <1606
bioReservoirs <1607
Model Blocks <1608
Molecules <1609
Protein Components <-1617
bioModels <-1618
Cytokines-Hormones <-1611
Receptors <-1612
Enzymes <-1613
Transcription Factors <-1614
Protein-DNA Comple <-1615
Other Proteins <-1616
1619
1621
Help & Mode Menus
Palettes
Structure Libraries
Cytokine Library
Enzyme Library
Receptor Library
TranscriptionF Library
Complex Library
Cytokine Library
Viral Protein Library
IL1.like Library
IL4.like Library<1622
IL6.like Library<1625
IL7.like Library
GH.like Library<1626
LIF.like Library
Other Hematopoietins
Other Cytokines
1620->
IL4.like LIBRARY
<-1623
'IL3'
'IL5'
'GM.CSF'
<-1624
'IL4'
'IL13'
'm.IL3'
'm.IL5'
'm.GM.CSF'
1628
Help & Mode Menus
Palettes
Structure Libraries
Pathway Libraries
Insulin.R.Pathway
EGF.R.Pathway
PDGF.R.Pathway
IFNa.R.Pathway
IFNg.R.Pathway
ECGF.R.Pathway
IL2.R.Pathway
IL4.R.Pathway
Other Pathways
1629->
'EGF.R.Pathway'
<-1632
<-1631
'EGF-EGF.R'
'EGF-di.EGF.R.TyrP_p120.GAP'
'EGF-di.EGF.R.TyrP_GRB2-SOS1_p120.GAP'
'mod-EGF-di.EGF.R.TyrP_GRB2-SOS1_p120.GAP + H.Ras.GTP-Raf1'
'EGF-di.EGF.R.tyrP'
'EGF-di.EGF.R.TyrP_p120.GAP.H.Ras.GTP'
'EGF-di.EGF.R_GRB2-SOS1_p120.GAP-H.Ras.GTP'
'EGF+EGF.R'
'EGF-di.EGF.R.TyrP_GRB2-SOS1-H.Ras.GDP_p120.GAP'
'EGF-di.EGF.R.TyrP-PLC.g1.tyr783P'
'mod-EGF-di.EGF.R.TyrP_p120.GAP + H.Ras.GTP-Raf1'
'EGF-di.EGF.R.tyrP Jak1.tyrP'
<-1630

FIG.17



a leucin-zipper-motif	
Names	LEUCIN.ZIPPER.MOTIF.MC
Label	"leucin.zipper"
Description	""
References	""
Id	""
Master bioentity	""
Superior bioentity	none
Position	""
Toggle state	hide
Toggle state 2	hide

a loss.of.function protein.mutation	
Names	LOSS.OF.FUNCTION.MUTATION.MC
Label	"loss.of.function.mutation"
Description	""
References	""
Id	""
Master bioentity	""
Superior bioentity	none
Position	""
Known mutagens	""

FIG.18

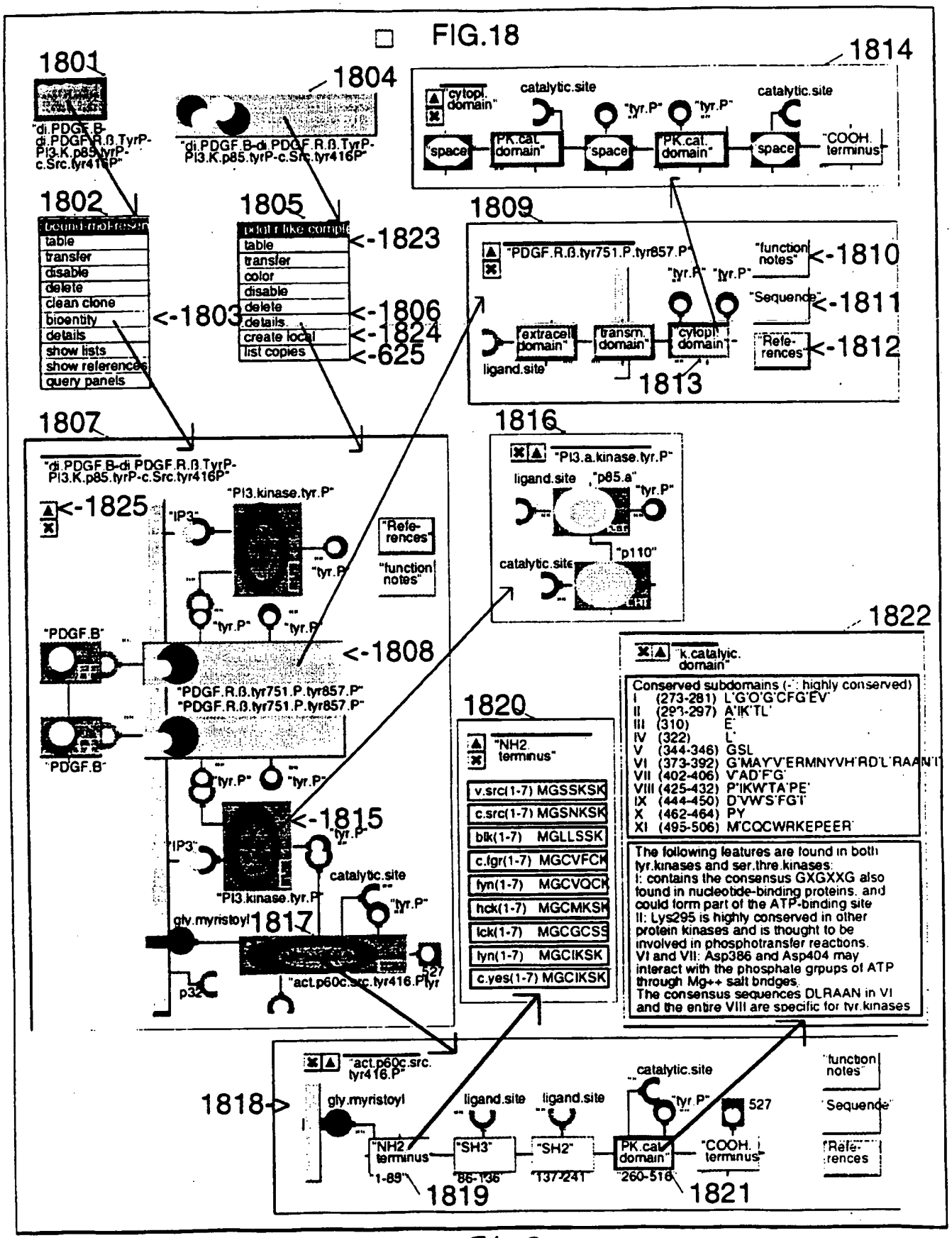

1801
1804
1814
"di.PDGF.B-di.PDGF.R.ß.TyrP-PI3-K.p85.tyrP-c.Src.tyr416P"
"di.PDGF.B-di.PDGF.R.ß.TyrP-PI3.K.p85.tyrP-c.Src.tyr416P"
cytopl. domain
catalytic.site
"tyr.P"
"tyr.P"
catalytic.site
space
PK.cat. domain
space
PK.cat. domain
space
COOH. terminus
1802
1805
1809
table
transfer
disable
delete
clean clone
bioentity
details
show lists
show references
query panels
<-1803
table
transfer
color
disable
delete
details
create local
list copies
<-1823
<-1806
<-1824
<-625
"PDGF.R.ß.tyr751.P.tyr857.P"
"function notes"
<-1810
"tyr.P"
"tyr.P"
"Sequence"
<-1811
"extracell. domain"
transm. domain
cytopl. domain
"Refe-rences"
<-1812
ligand.site
1813
1807
1816
"di.PDGF.B-di.PDGF.R.ß.TyrP-PI3.K.p85.tyrP-c.Src.tyr416P"
<-1825
"PI3.kinase.tyr.P"
"IP3"
"tyr.P"
"Refe-rences"
"function notes"
"tyr.P"
"tyr.P"
"PI3.a.kinase.tyr.P"
ligand.site
"p85.a"
"tyr.P"
"p110"
catalytic.site
"PDGF.B"
<-1808
"PDGF.R.ß.tyr751.P.tyr857.P"
"PDGF.R.ß.tyr751.P.tyr857.P"
"PDGF.B"
"tyr.P"
"tyr.P"
1822
1820
<-1815
"tyr.P"
"IP3"
catalytic.site
"PI3.kinase.tyr.P"
"tyr.P"
gly.myristoyl
1817
"act.p60c.src.tyr416.P"tyr
527
p32
"k.catalyic. domain"
Conserved subdomains (-: highly conserved)
I (273-281) L'G'O'G'CFG'EV
II (293-297) A'IK'TL'
III (310) E
IV (322) L
V (344-346) GSL
VI (373-392) G'MAY'V'ERMNYVH'RD'L'RAAN'I
VII (402-406) V'AD'F'G'
VIII (425-432) P'IKW'TA'PE'
IX (444-450) D'VW'S'FG'I
X (462-464) PY
XI (495-506) M'CQCWRKEPEER'
"NH2 terminus"
v.src(1-7) MGSSKSK
c.src(1-7) MGSNKSK
blk(1-7) MGLLSSK
c.fgr(1-7) MGCVFCK
fyn(1-7) MGCVQCK
hck(1-7) MGCMKSK
lck(1-7) MGCGCSS
lyn(1-7) MGCIKSK
c.yes(1-7) MGCIKSK
The following features are found in both tyr.kinases and ser.thre.kinases:
I: contains the consensus GXGXXG also found in nucleotide-binding proteins, and could form part of the ATP-binding site
II: Lys295 is highly conserved in other protein kinases and is thought to be involved in phosphotransfer reactions.
VI and VII: Asp386 and Asp404 may interact with the phosphate grpups of ATP through Mg++ salt bndges.
The consensus sequences DLRAAN in VI and the entire VIII are specific for tyr.kinases
"act.p60c.src. tyr416.P"
1818
gly.myristoyl
ligand.site
ligand.site
catalytic.site
"tyr.P"
527
"function notes"
"Sequence"
"Refe-rences"
"NH2 terminus"
"SH3"
"SH2"
PK.cat. domain
"COOH. terminus"
1-89
86-136
137-241
260-516
1819
1821

FIG.19

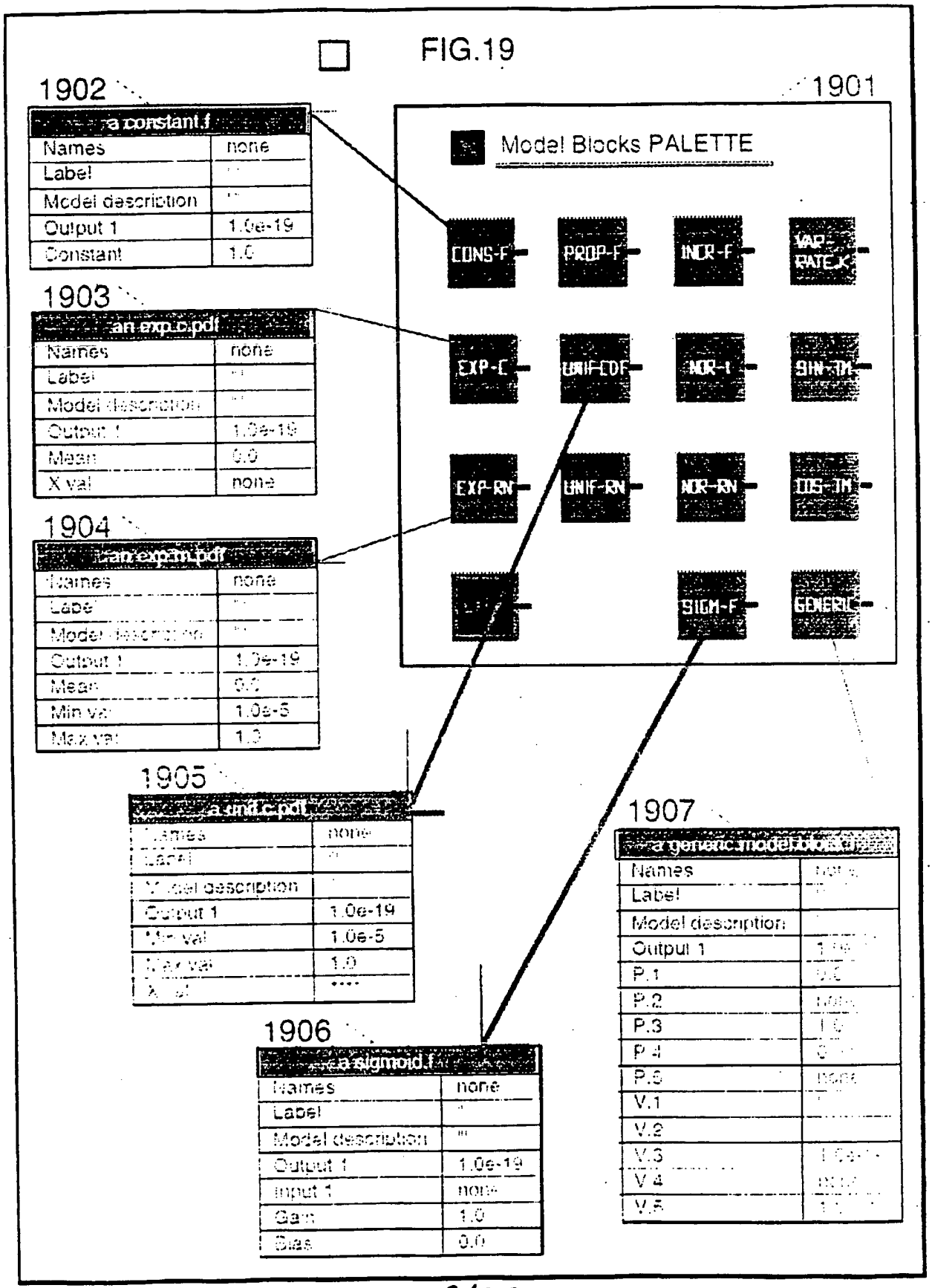

1902
a constant.f
Names none
Label
Model description
Output 1 1.0e-19
Constant 1.0
1903
an exp c.pdf
Names none
Label
Model description
Output 1 1.0e-19
Mean 0.0
X val none
1904
an exp m.pdf
Names none
Label
Model description
Output 1 1.0e-19
Mean 0.0
Min val 1.0e-5
Max val 1.0
1905
a unif c.pdf
Names none
Label
Model description
Output 1 1.0e-19
Min val 1.0e-5
Max val 1.0
1906
a sigmoid.f
Names none
Label
Model description
Output 1 1.0e-19
Input 1 none
Gain 1.0
Bias 0.0
1907
a generic model.blocks
Names
Label
Model description
Output 1
P.1
P.2
P.3
P.4
P.5
V.1
V.2
V.3
V.4
V.5
1901
Model Blocks PALETTE
CONS-F
PROP-F
INCR-F
VAR-RATE-K
EXP-C
UNIF-CDF
NOR-I
SIN-TM
EXP-RN
UNIF-RN
NOR-RN
DIS-TM
SIGM-F
GENERIC

FIG.20

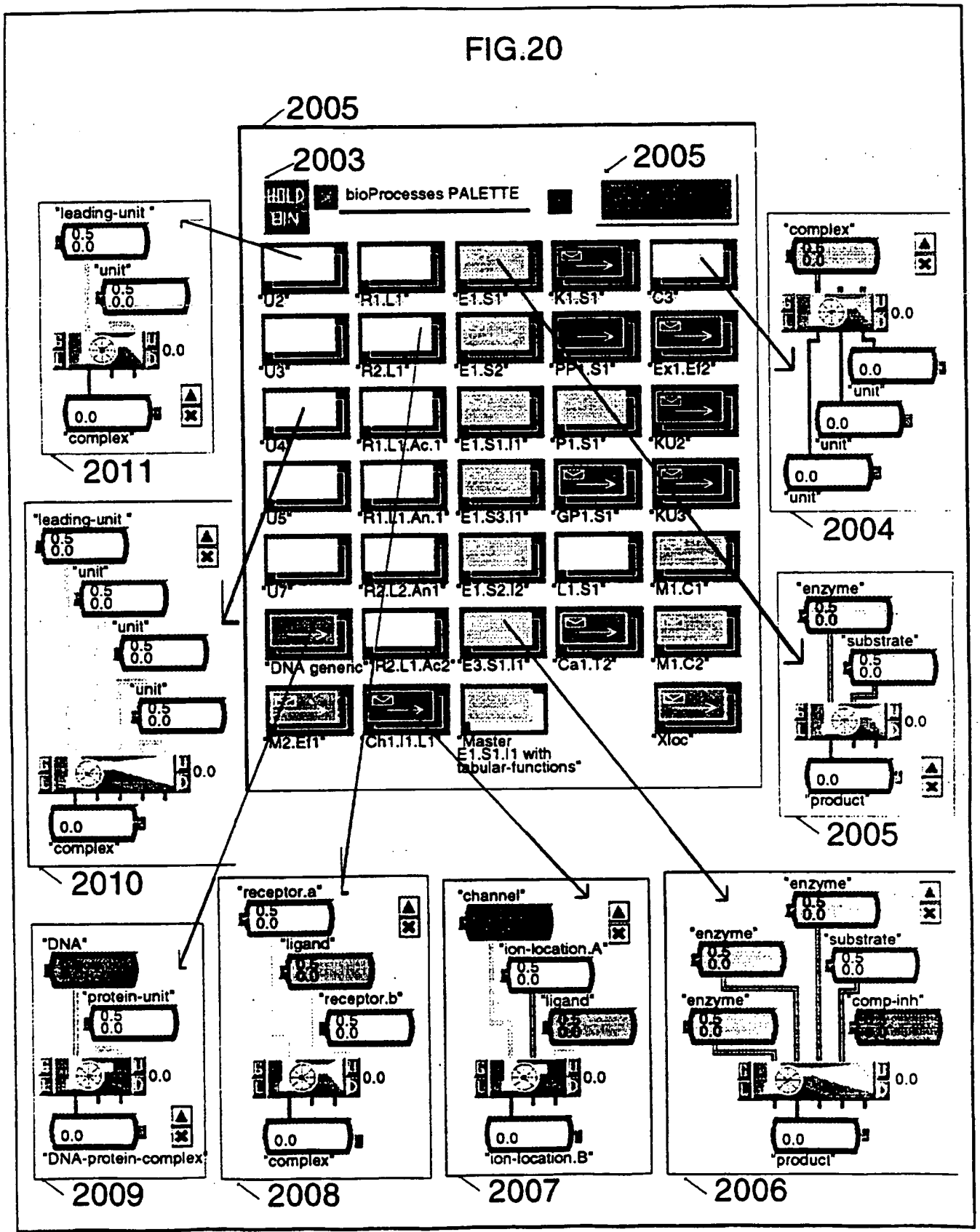

2005
2003
2005
bioProcesses PALETTE
"leading-unit "
"unit"
"complex"
2011
"U2" "R1.L1" "E1.S1" "K1.S1" "C3"
"U3" "R2.L1" "E1.S2" "PP1.S1" "Ex1.E/2"
"U4" "R1.L1.Ac.1" "E1.S1.I1" "P1.S1" "KU2"
"U5" "R1.L1.An.1" "E1.S3.I1" "GP1.S1" "KU3"
"U7" "R2.L2.An1" "E1.S2.I2" "L1.S1" "M1.C1"
"DNA generic" "R2.L1.Ac2" "E3.S1.I1" "Ca1.T2" "M1.C2"
"M2.E/1" "Ch1.I1.L1" "Master E1.S1.I1 with tabular-functions" "Xloc"
"complex"
"unit"
"unit"
2004
"leading-unit "
"unit"
"unit"
"unit"
"complex"
2010
"enzyme"
"substrate"
"product"
2005
"DNA"
"protein-unit"
"DNA-protein-complex"
2009
"receptor.a"
"ligand"
"receptor.b"
"complex"
2008
"channel"
"ion-location.A"
"ligand"
"ion-location.B"
2007
"enzyme"
"enzyme"
"substrate"
"enzyme"
"comp-inh"
"product"
2006

FIG.21

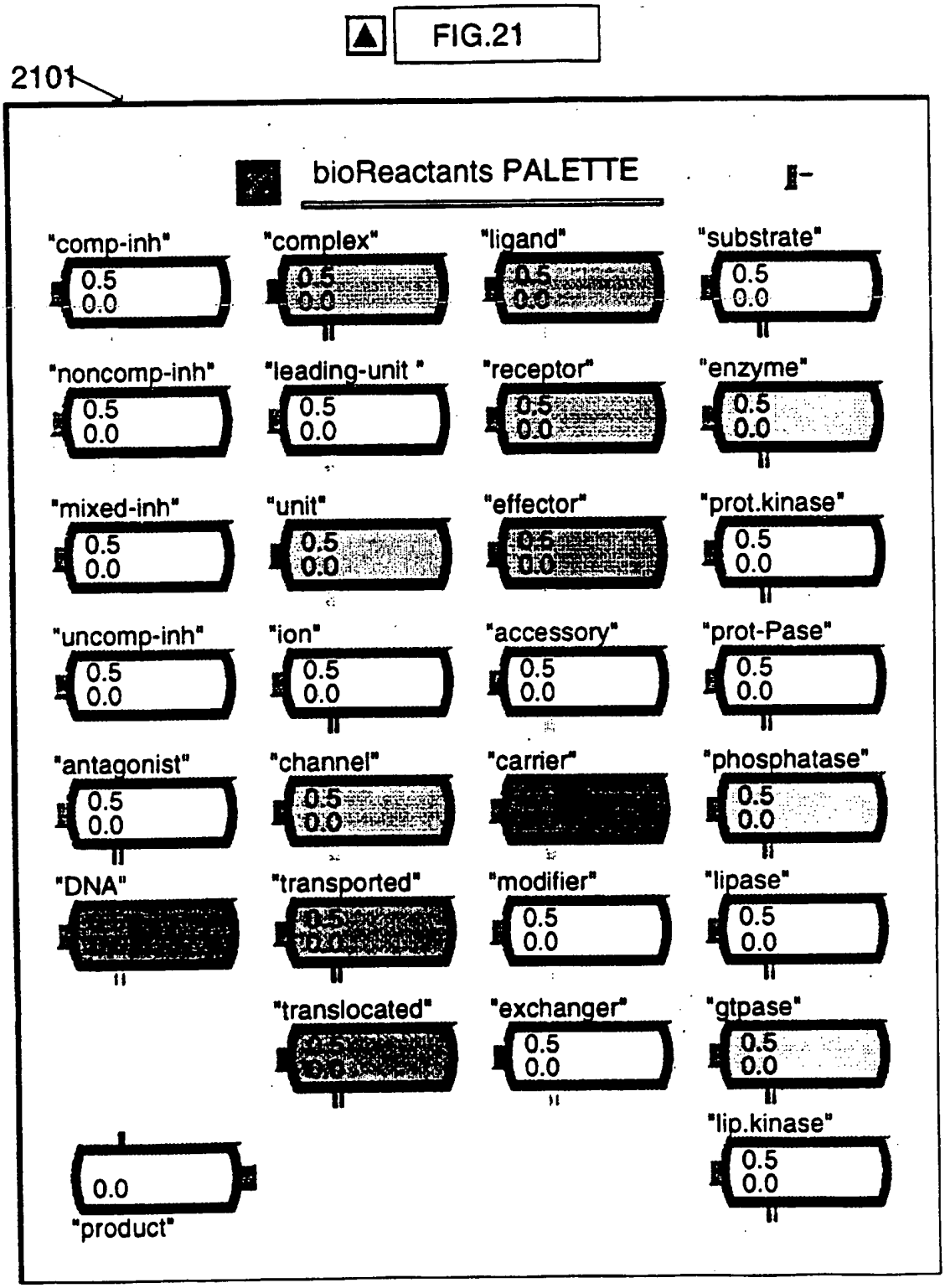

2101
bioReactants PALETTE
"comp-inh"
"complex"
"ligand"
"substrate"
"noncomp-inh"
"leading-unit "
"receptor"
"enzyme"
"mixed-inh"
"unit"
"effector"
"prot.kinase"
"uncomp-inh"
"ion"
"accessory"
"prot-Pase"
"antagonist"
"channel"
"carrier"
"phosphatase"
"DNA"
"transported"
"modifier"
"lipase"
"translocated"
"exchanger"
"gtpase"
"lip.kinase"
"product"
0.5
0.0

FIG.22

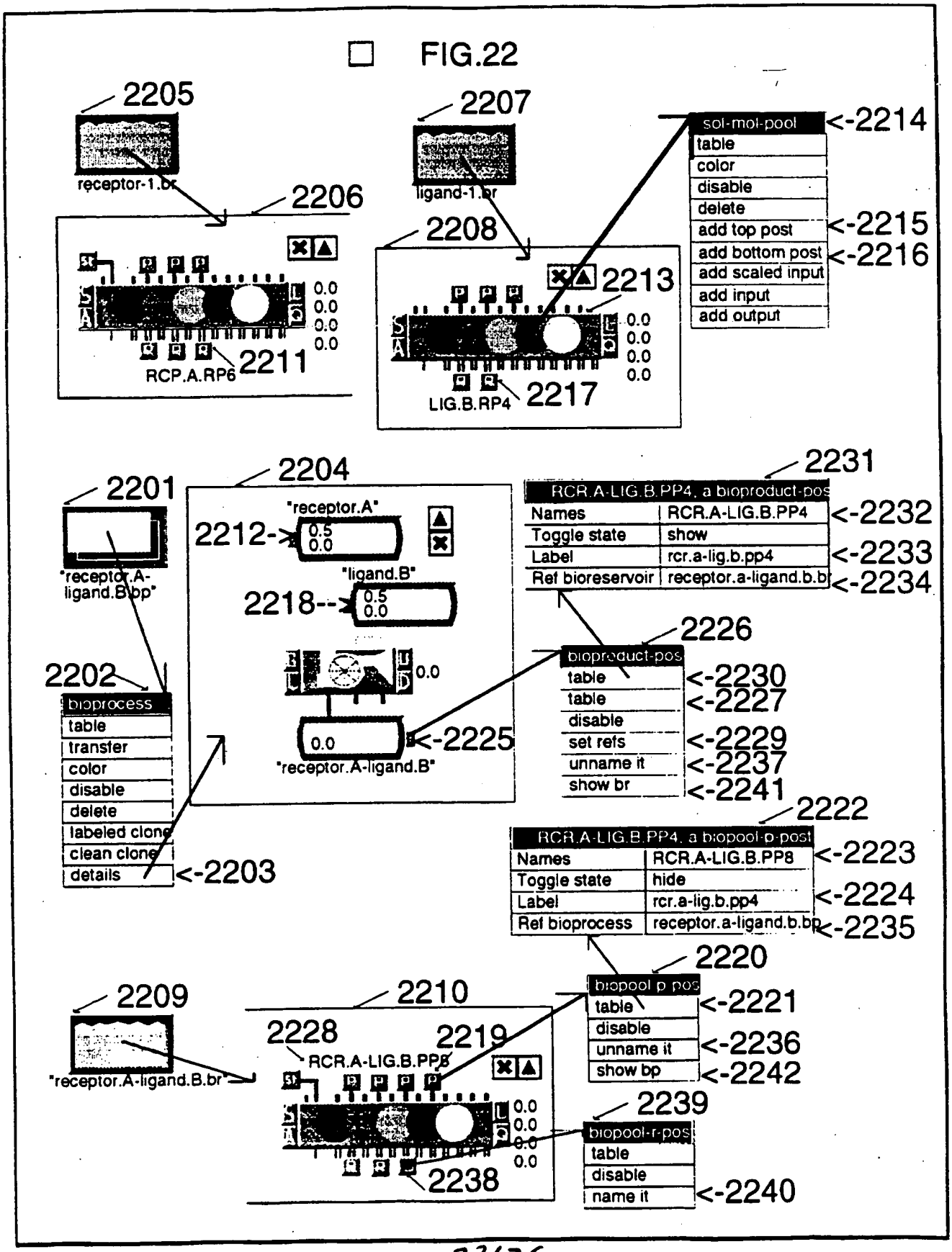

2205
receptor-1.br
2207
ligand-1.br
2206
2208
sol-mol-pool
table
color
disable
delete
add top post
add bottom post
add scaled input
add input
add output
<-2214
<-2215
<-2216
2213
0.0
0.0
0.0
0.0
RCP.A.RP6
2211
LIG.B.RP4
2217
2201
"receptor.A-ligand.B.bp"
2204
"receptor.A"
2212->
0.5
0.0
"ligand.B"
2218-->
0.5
0.0
0.0
0.0
<-2225
"receptor.A-ligand.B"
2231
RCR.A-LIG.B.PP4, a bioproduct-pos
Names
RCR.A-LIG.B.PP4
Toggle state
show
Label
rcr.a-lig.b.pp4
Ref bioreservoir
receptor.a-ligand.b.b
<-2232
<-2233
<-2234
2226
bioproduct-pos
table
table
disable
set refs
unname it
show br
<-2230
<-2227
<-2229
<-2237
<-2241
2202
bioprocess
table
transfer
color
disable
delete
labeled clone
clean clone
details
<-2203
2222
RCR.A-LIG.B.PP4, a biopool-p-post
Names
RCR.A-LIG.B.PP8
Toggle state
hide
Label
rcr.a-lig.b.pp4
Ref bioprocess
receptor.a-ligand.b.bp
<-2223
<-2224
<-2235
2220
biopool p pos
table
disable
unname it
show bp
<-2221
<-2236
<-2242
2209
"receptor.A-ligand.B.br"
2210
2228
RCR.A-LIG.B.PP8
2219
0.0
0.0
0.0
0.0
2238
2239
biopool-r-pos
table
disable
name it
<-2240

FIG.23

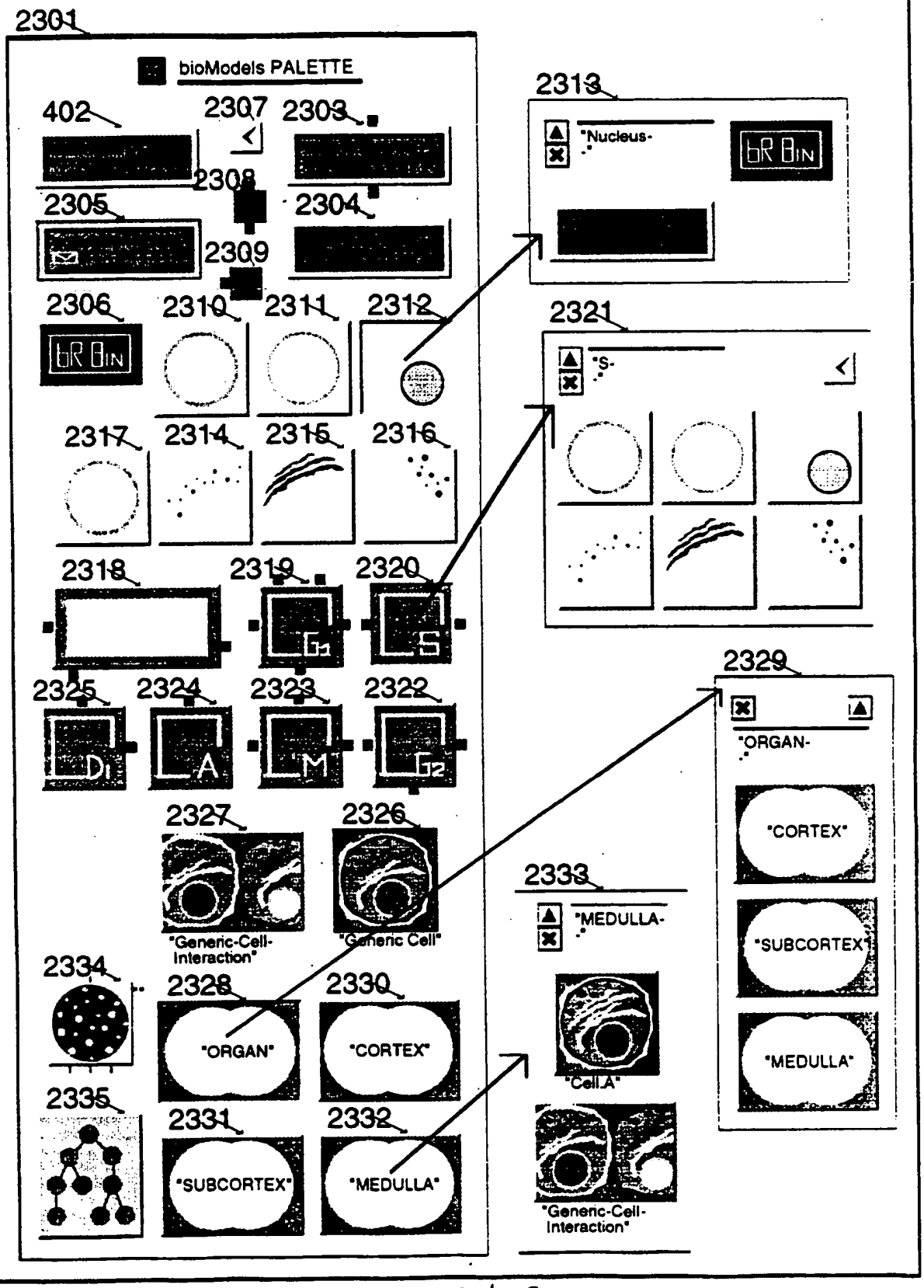

2301
bioModels PALETTE
402
2307
2303
2308
2305
2304
2309
2306
2310
2311
2312
2317
2314
2315
2316
2318
2319
2320
2325
2324
2323
2322
2327
2326
"Generic-Cell-Interaction"
"Generic Cell"
2334
2328
2330
"ORGAN"
"CORTEX"
2335
2331
2332
"SUBCORTEX"
"MEDULLA"
2313
"Nucleus-
2321
"S-
2329
"ORGAN-
"CORTEX"
"SUBCORTEX"
"MEDULLA"
2333
"MEDULLA-
"Cell A"
"Generic-Cell-Interaction"

FIG.24

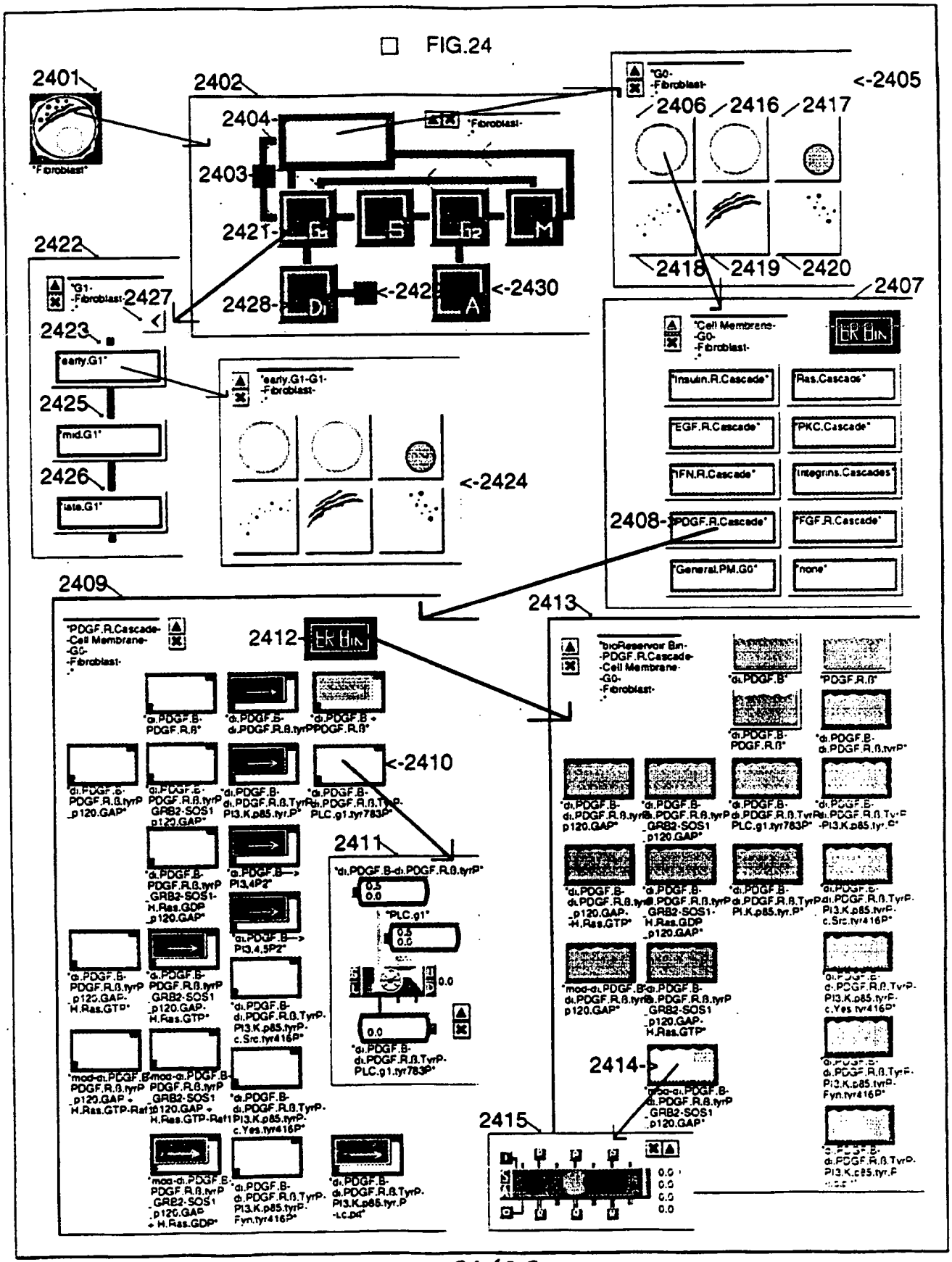

2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2430
Fibroblast
G0-Fibroblast
G1-Fibroblast
early.G1
mid.G1
late.G1
early.G1-G1-Fibroblast
Cell Membrane-G0-Fibroblast
Insulin.R.Cascade
Ras.Cascade
EGF.R.Cascade
PKC.Cascade
IFN.R.Cascade
Integrins.Cascades
PDGF.R.Cascade
FGF.R.Cascade
General.PM.G0
none
PDGF.R.Cascade-Cell Membrane-G0-Fibroblast
bioReservoir Bin-PDGF.R.Cascade-Cell Membrane-G0-Fibroblast
di.PDGF.B-PDGF.R.ß
di.PDGF.B-di.PDGF.R.ß.tyrP
di.PDGF.B-PDGF.R.ß
di.PDGF.B-PDGF.R.ß.tyrP_p120.GAP
di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1_p120.GAP
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyr.P
di.PDGF.B-di.PDGF.R.ß.TyrP-PLC.g1.tyr783P
di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1-H.Ras.GDP_p120.GAP
di.PDGF.B-PI3.4P2
di.PDGF.B-PI3.4.5P2
di.PDGF.B-PDGF.R.ß.tyrP_p120.GAP-H.Ras.GTP
di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1_p120.GAP-H.Ras.GTP
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-c.Src.tyr416P
mod-di.PDGF.B-PDGF.R.ß.tyrP_p120.GAP+H.Ras.GTP-Raf1
mod-di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1_p120.GAP+H.Ras.GTP-Raf1
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-c.Yes.tyr416P
mod-di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1_p120.GAP+H.Ras.GDP
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-Fyn.tyr416P
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyr.P-Lc.pic
di.PDGF.B-di.PDGF.R.ß.tyrP
"PLC.g1"
di.PDGF.B-di.PDGF.R.ß.TyrP-PLC.g1.tyr783P
mod-di.PDGF.B-PDGF.R.ß.tyrP_GRB2-SOS1_p120.GAP-H.Ras.GTP
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-c.Src.tyr416P
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-c.Yes.tyr416P
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyrP-Fyn.tyr416P
di.PDGF.B-di.PDGF.R.ß.TyrP-PI3K.p85.tyr.P
0.5
0.0
0.0
0.0
0.0
0.0
0.0
0.0

FIG.25

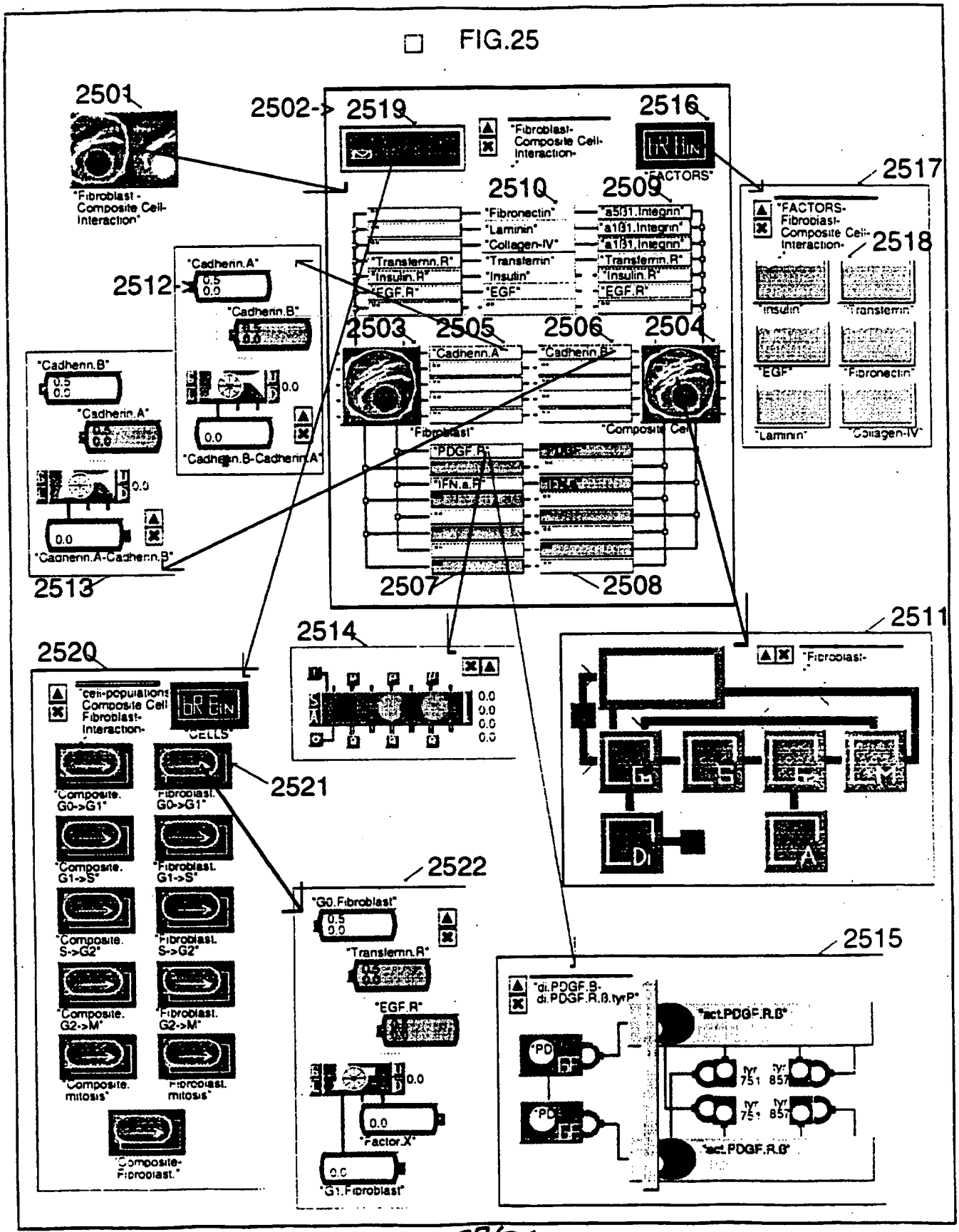

2501
"Fibroblast - Composite Cell-Interaction"
2502
2519
"Fibroblast-Composite Cell-Interaction-"
2516
"FACTORS"
2517
"FACTORS-Fibroblast-Composite Cell-Interaction-"
2518
"Insulin"
"Transferrin"
"EGF"
"Fibronectin"
"Laminin"
"Collagen-IV"
2510
2509
"Fibronectin"
"Laminin"
"Collagen-IV"
"Transfernn.R"
"Insulin.R"
"EGF.R"
"Transferrin"
"Insulin"
"EGF"
"a5β1.Integrin"
"a1β1.Integrin"
"a1β1.Integrin"
2512
"Cadherin.A"
"Cadherin.B"
"Cadherin.B-Cadherin.A"
2503
2505
2506
2504
"Cadherin.A"
"Cadherin.B"
"Fibroblast"
"Composite Cell"
"PDGF.R"
"IFN.a.R"
"Cadherin.B"
"Cadherin.A"
"Cadherin.A-Cadherin.B"
2513
2507
2508
2514
2511
"Fibroblast-"
2520
"cell-populations Composite Cell Fibroblast-Interaction-"
"CELLS"
"Composite. G0->G1"
"Fibroblast. G0->G1"
"Composite. G1->S"
"Fibroblast. G1->S"
"Composite. S->G2"
"Fibroblast. S->G2"
"Composite. G2->M"
"Fibroblast. G2->M"
"Composite. mitosis"
"Fibroblast. mitosis"
"Composite-Fibroblast."
2521
2522
"G0.Fibroblast"
"Transfernn.R"
"EGF.R"
"Factor.X"
"G1.Fibroblast"
2515
"di.PDGF.B-di.PDGF.R.B.tyrP"
"act.PDGF.R.B"
"PDGF"
"PDGF"
tyr 751
tyr 857
tyr 751
tyr 857
"act.PDGF.R.B"

FIG.26

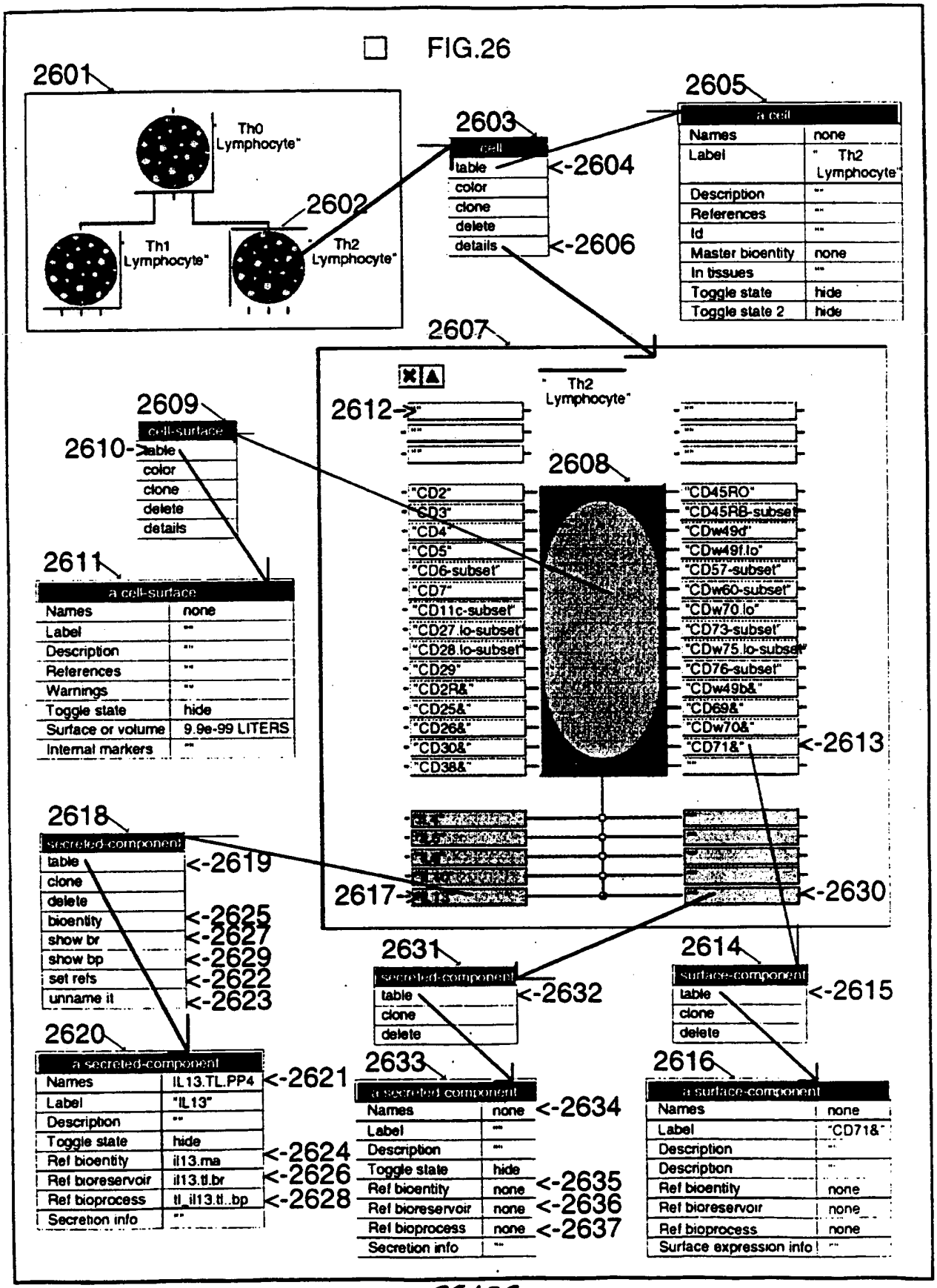

2601
Th0 Lymphocyte"
Th1 Lymphocyte"
Th2 Lymphocyte"
2602
2603
cell
table
color
clone
delete
details
<-2604
<-2606
2605
a cell
Names none
Label " Th2 Lymphocyte"
Description ""
References ""
Id ""
Master bioentity none
In tissues ""
Toggle state hide
Toggle state 2 hide
2607
Th2 Lymphocyte"
2608
"CD2"
"CD3"
"CD4"
"CD5"
"CD6-subset"
"CD7"
"CD11c-subset"
"CD27.lo-subset"
"CD28.lo-subset"
"CD29"
"CD2R&"
"CD25&"
"CD26&"
"CD30&"
"CD38&"
"CD45RO"
"CD45RB-subset"
"CDw49d"
"CDw49f.lo"
"CD57-subset"
"CDw60-subset"
"CDw70.lo"
"CD73-subset"
"CDw75.lo-subset"
"CD76-subset"
"CDw49b&"
"CD69&"
"CDw70&"
"CD71&"
<-2613
2609
cell-surface
table
color
clone
delete
details
2610
2611
a cell-surface
Names none
Label ""
Description ""
References ""
Warnings ""
Toggle state hide
Surface or volume 9.9e-99 LITERS
Internal markers ""
2612
2617
<-2630
2618
secreted-component
table
clone
delete
bioentity
show br
show bp
set refs
unname it
<-2619
<-2625
<-2627
<-2629
<-2622
<-2623
2620
a secreted-component
Names IL13.TL.PP4
Label "IL13"
Description ""
Toggle state hide
Ref bioentity il13.ma
Ref bioreservoir il13.tl.br
Ref bioprocess tl_il13.tl..bp
Secretion info ""
<-2621
<-2624
<-2626
<-2628
2631
secreted-component
table
clone
delete
<-2632
2633
a secreted component
Names none
Label ""
Description ""
Toggle state hide
Ref bioentity none
Ref bioreservoir none
Ref bioprocess none
Secretion info ""
<-2634
<-2635
<-2636
<-2637
2614
surface-component
table
clone
delete
<-2615
2616
a surface-component
Names none
Label "CD71&"
Description ""
Description ""
Ref bioentity none
Ref bioreservoir none
Ref bioprocess none
Surface expression info ""

FIG.27

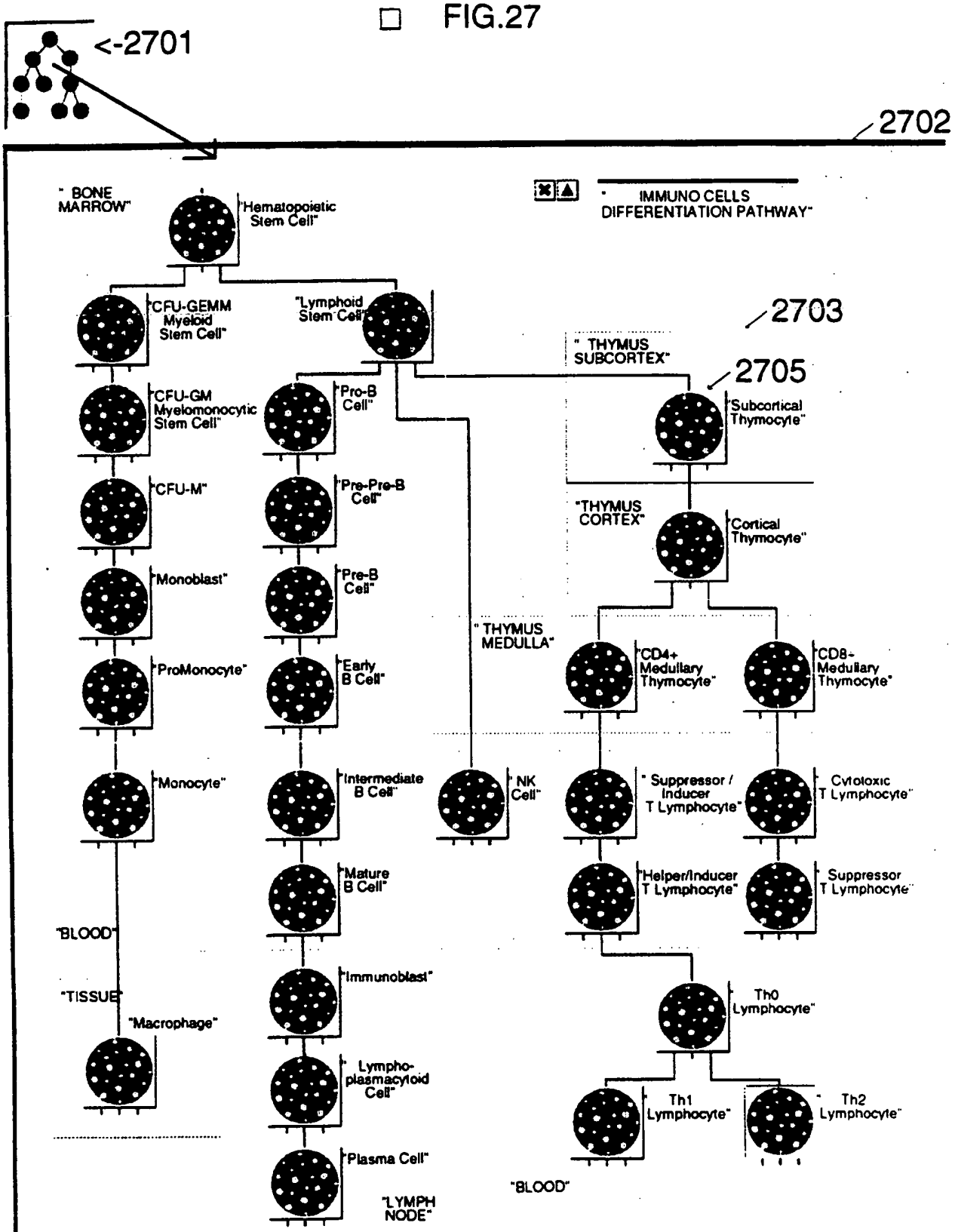

<-2701
2702
" BONE MARROW"
"Hematopoietic Stem Cell"
" IMMUNO CELLS DIFFERENTIATION PATHWAY"
"CFU-GEMM Myeloid Stem Cell"
"Lymphoid Stem Cell"
2703
" THYMUS SUBCORTEX"
2705
"CFU-GM Myelomonocytic Stem Cell"
"Pro-B Cell"
"Subcortical Thymocyte"
"CFU-M"
"Pre-Pre-B Cell"
"THYMUS CORTEX"
"Cortical Thymocyte"
"Monoblast"
"Pre-B Cell"
" THYMUS MEDULLA"
"CD4+ Medullary Thymocyte"
"CD8- Medullary Thymocyte"
"ProMonocyte"
"Early B Cell"
"Monocyte"
"Intermediate B Cell"
"NK Cell"
" Suppressor / Inducer T Lymphocyte"
" Cytotoxic T Lymphocyte"
"Mature B Cell"
"Helper/Inducer T Lymphocyte"
" Suppressor T Lymphocyte"
"BLOOD"
"Immunoblast"
"TISSUE"
"Macrophage"
"Th0 Lymphocyte"
" Lympho-plasmacytoid Cell"
"Th1 Lymphocyte"
" Th2 Lymphocyte"
"Plasma Cell"
"BLOOD"
"LYMPH NODE"

FIG.28

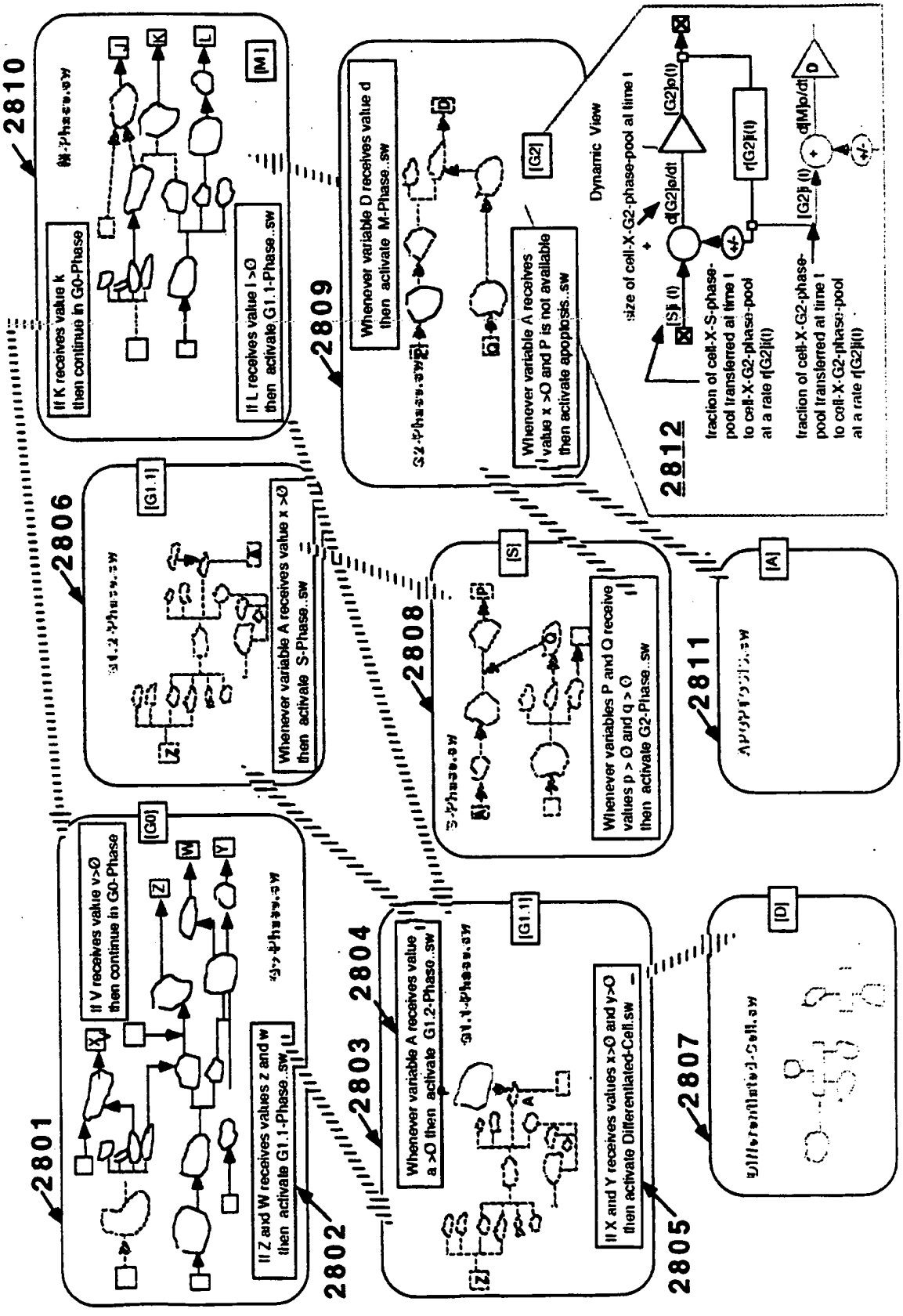

2810
2809
2806
2808
2811
2801
2803
2804
2802
2805
2807
2812
Dynamic View
If K receives value k then continue in G0-Phase
If L receives value l >0 then activate G1.1-Phase..sw
Whenever variable D receives value d then activate M-Phase..sw
Whenever variable A receives value x >0 and P is not available then activate apoptosis..sw
Whenever variable A receives value x >0 then activate S-Phase..sw
Whenever variables P and Q receive values p > 0 and q > 0 then activate G2-Phase..sw
If V receives value v>0 then continue in G0-Phase
If Z and W receives values z and w then activate G1.1-Phase..sw
Whenever variable A receives value a >0 then activate G1.2-Phase..sw
If X and Y receives values x>0 and y>0 then activate Differentiated-Cell.sw
size of cell-X-G2-phase-pool at time t
fraction of cell-X-S-phase-pool transfered at time t to cell-X-G2-phase-pool at a rate r[G2]i(t)
fraction of cell-X-G2-phase-pool transfered at time t to cell-X-G2-phase-pool at a rate r[G2]i(t)

FIG.29

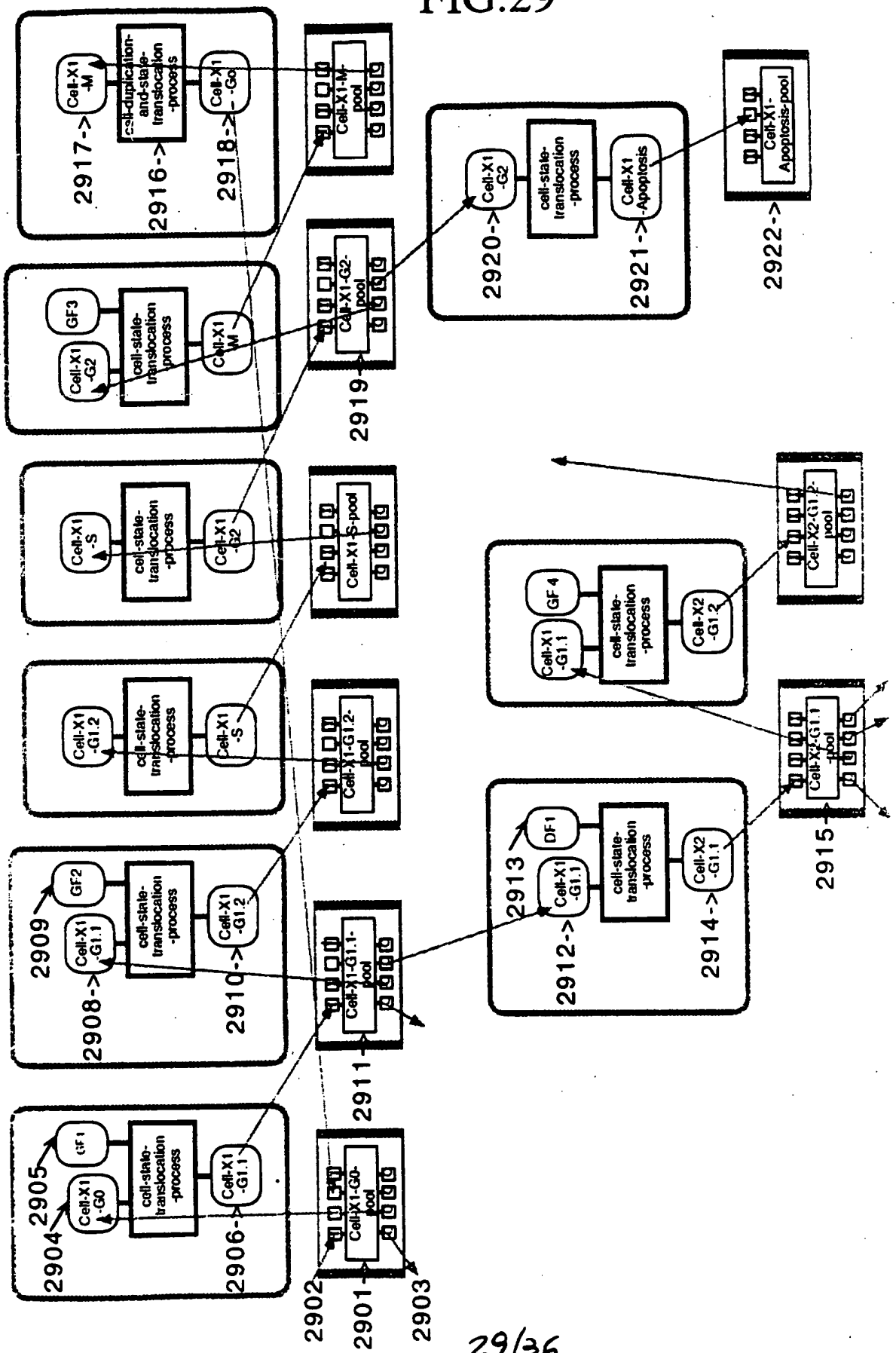

2901
2902
2903
2904
2905
2906->
2908->
2909
2910->
2911
2912->
2913
2914->
2915
2916->
2917->
2918->
2919
2920->
2921->
2922->
Cell-X1 -G0
GF1
cell-state-translocation -process
Cell-X1 -G1.1
Cell-X1-G0-pool
Cell-X1-G1.1-pool
Cell-X1 -G1.2
Cell-X1-G1.2-pool
Cell-X1 -S
Cell-X1-S-pool
Cell-X1 -G2
Cell-X1-G2-pool
Cell-X1 -M
GF3
Cell-X1-M-pool
GF2
DF1
Cell-X1 -G1.1
Cell-X2 -G1.1
Cell-X2-G1.1-pool
GF4
Cell-X2 -G1.2
Cell-X2-G1.2-pool
cell-duplication-and-state-translocation -process
Cell-X1 -G2
Cell-X1 -Apoptosis
Cell-X1-Apoptosis-pool

FIG.30

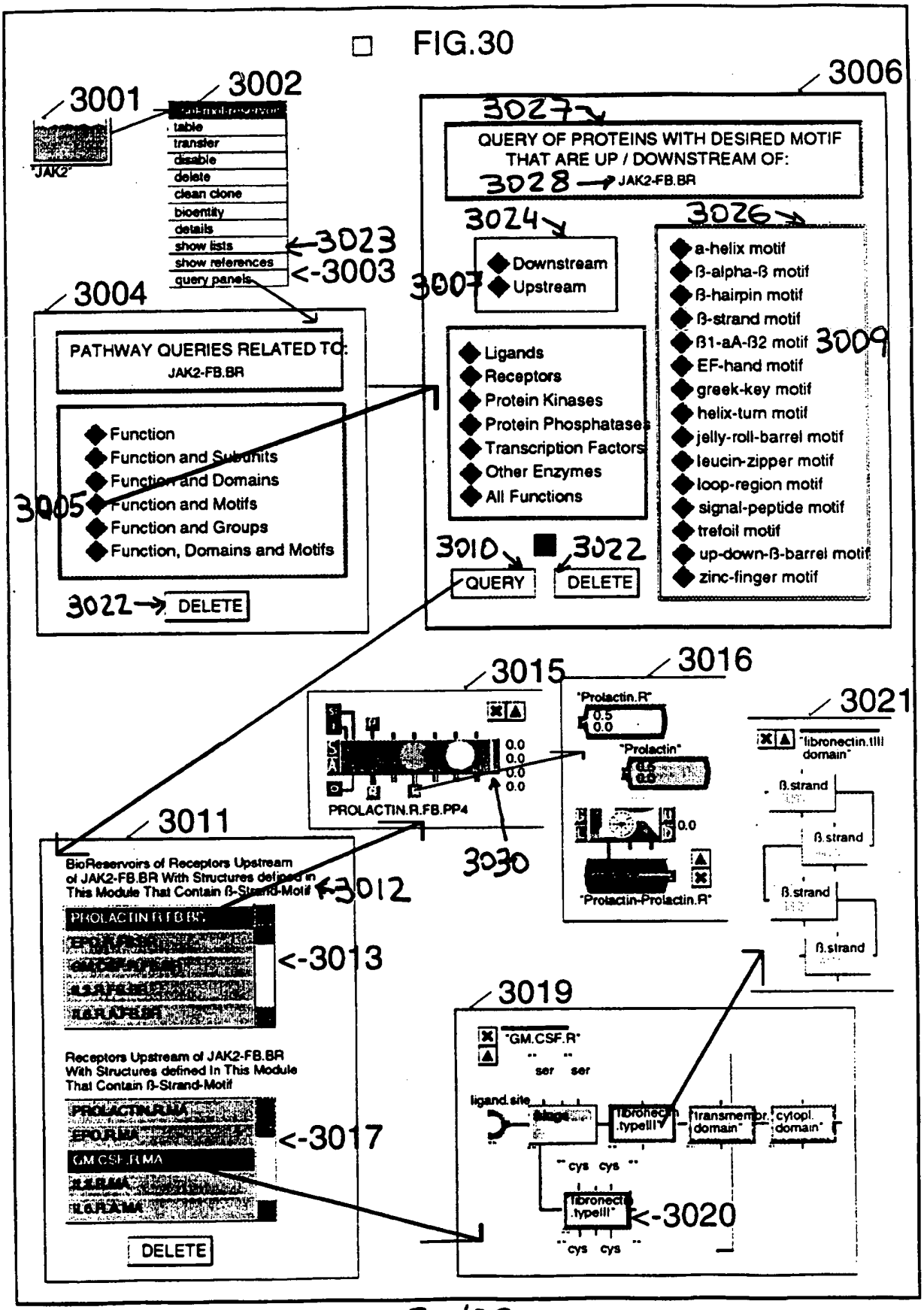

3001
3002
"JAK2"
table
transfer
disable
delete
clean clone
bioentity
details
show lists
show references
query panels
<-3023
<-3003
3004
PATHWAY QUERIES RELATED TO:
JAK2-FB.BR
Function
Function and Subunits
Function and Domains
Function and Motifs
Function and Groups
Function, Domains and Motifs
3005
3022
DELETE
3006
3027
QUERY OF PROTEINS WITH DESIRED MOTIF
THAT ARE UP / DOWNSTREAM OF:
3028
JAK2-FB.BR
3024
Downstream
Upstream
3007
3026
a-helix motif
ß-alpha-ß motif
ß-hairpin motif
ß-strand motif
ß1-aA-ß2 motif
EF-hand motif
greek-key motif
helix-turn motif
jelly-roll-barrel motif
leucin-zipper motif
loop-region motif
signal-peptide motif
trefoil motif
up-down-ß-barrel motif
zinc-finger motif
3009
Ligands
Receptors
Protein Kinases
Protein Phosphatases
Transcription Factors
Other Enzymes
All Functions
3010
QUERY
3022
DELETE
3015
PROLACTIN.R.FB.PP4
3030
3016
"Prolactin.R"
"Prolactin"
"Prolactin-Prolactin.R"
3021
"fibronectin.III domain"
ß.strand
ß.strand
ß.strand
ß.strand
3011
BioReservoirs of Receptors Upstream of JAK2-FB.BR With Structures defined in This Module That Contain ß-Strand-Motif
3012
<-3013
Receptors Upstream of JAK2-FB.BR With Structures defined In This Module That Contain ß-Strand-Motif
PROLACTIN.R.MA
EPO.R.MA
GM.CSF.R.MA
<-3017
DELETE
3019
"GM.CSF.R"
ser ser
ligand.site
"transmembr. domain"
"cytopl. domain"
cys cys
cys cys
<-3020

FIG.31

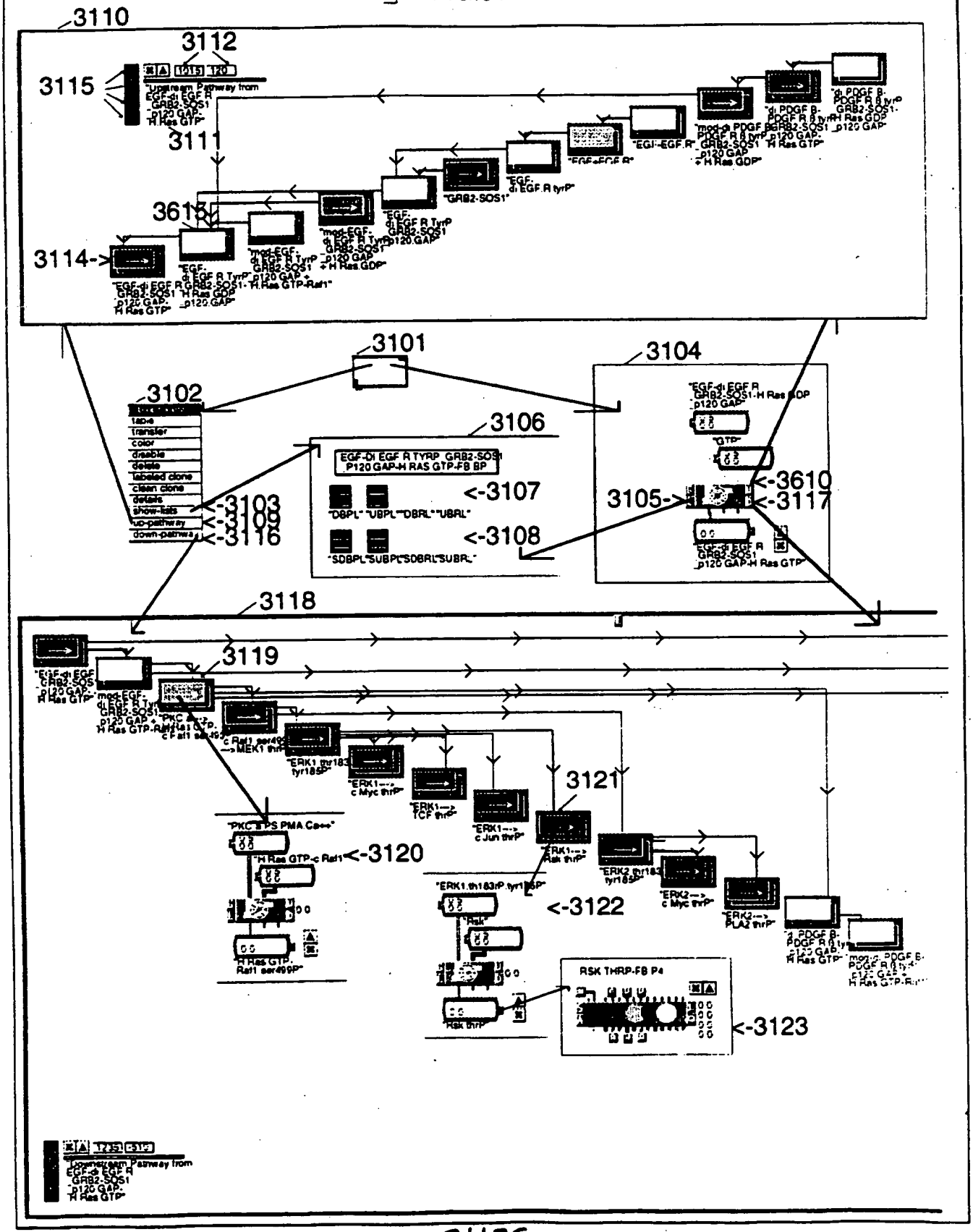

3110
3112
3115
3111
3613
3114->
3101
3102
3104
3106
EGF-di EGF R TYRP GRB2-SOS1 P120 GAP-H RAS GTP-FB BP
<-3107
<-3108
3105->
<-3610
<-3117
<-3103
<-3109
<-3116
3118
3119
3121
<-3120
<-3122
RSK THRP-FB P4
<-3123

FIG.32

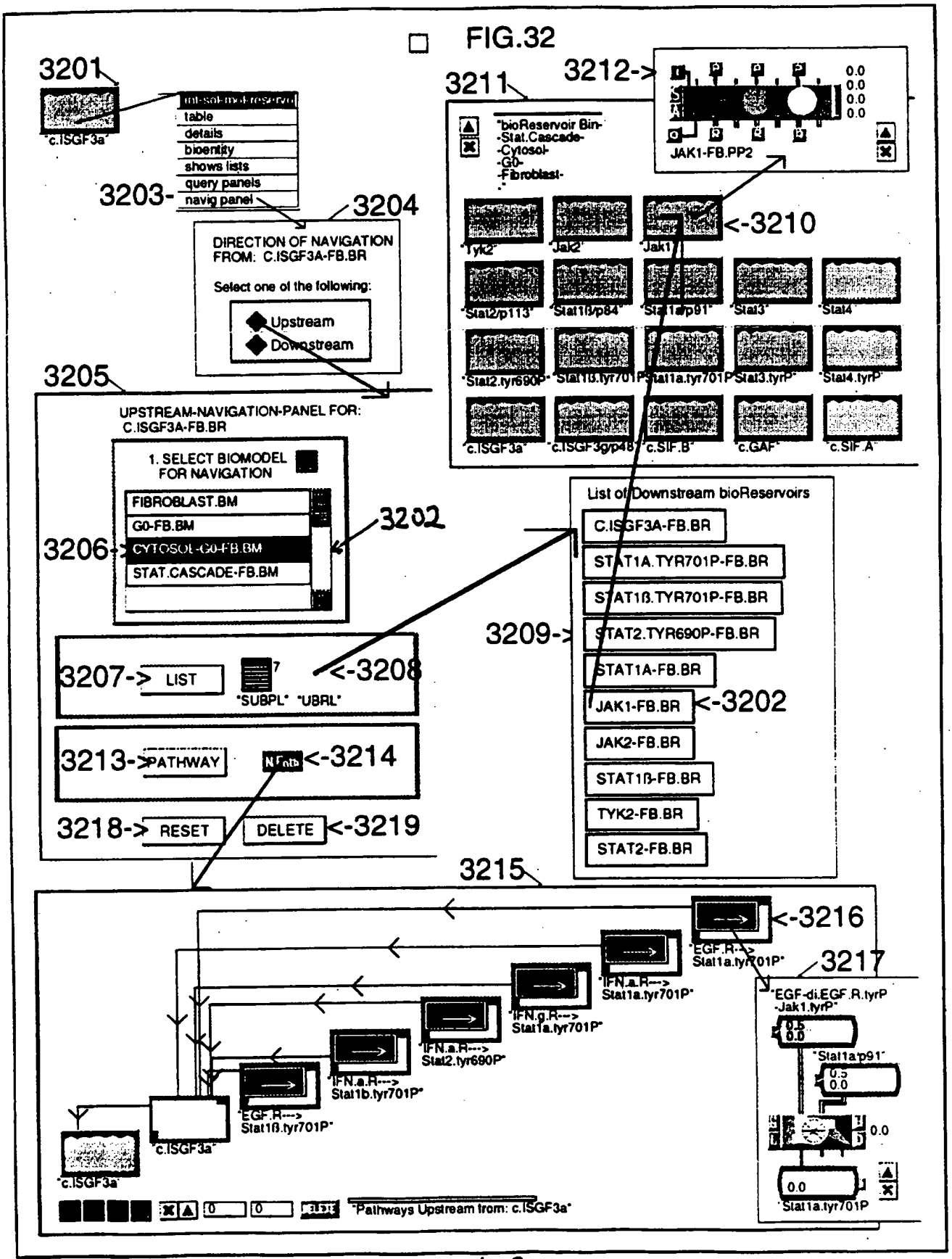

3201
c.ISGF3a
table
details
bioentity
shows lists
query panels
navig panel
3203-
3204
DIRECTION OF NAVIGATION
FROM: C.ISGF3A-FB.BR
Select one of the following:
Upstream
Downstream
3205
UPSTREAM-NAVIGATION-PANEL FOR:
C.ISGF3A-FB.BR
1. SELECT BIOMODEL
FOR NAVIGATION
FIBROBLAST.BM
G0-FB.BM
CYTOSOL-G0-FB.BM
STAT.CASCADE-FB.BM
3206->
3202
3207-> LIST
"SUBPL" "UBRL"
<-3208
3213-> PATHWAY
NFκb
<-3214
3218-> RESET
DELETE
<-3219
3211
3212->
"bioReservoir Bin-
-Stat.Cascade-
-Cytosol-
-G0-
-Fibroblast-
JAK1-FB.PP2
Tyk2
Jak2
Jak1
<-3210
Stat2/p113
Stat1ß/p84
Stat1a/p91
Stat3
Stat4
Stat2.tyr690P
Stat1ß.tyr701P
Stat1a.tyr701P
Stat3.tyrP
Stat4.tyrP
c.ISGF3a
c.ISGF3g/p48
c.SIF.B
c.GAF
c.SIF.A
List of Downstream bioReservoirs
C.ISGF3A-FB.BR
STAT1A.TYR701P-FB.BR
STAT1ß.TYR701P-FB.BR
3209->
STAT2.TYR690P-FB.BR
STAT1A-FB.BR
JAK1-FB.BR
<-3202
JAK2-FB.BR
STAT1ß-FB.BR
TYK2-FB.BR
STAT2-FB.BR
3215
<-3216
"EGF.R--->
Stat1a.tyr701P"
"IFN.a.R--->
Stat1a.tyr701P"
"IFN.g.R--->
Stat1a.tyr701P"
"IFN.a.R--->
Stat2.tyr690P"
"IFN.a.R--->
Stat1b.tyr701P"
"EGF.R--->
Stat1ß.tyr701P"
c.ISGF3a
c.ISGF3a
"Pathways Upstream from: c.ISGF3a"
3217
"EGF-di.EGF.R.tyrP
-Jak1.tyrP"
"Stat1a.p91"
"Stat1a.tyr701P"

FIG.33

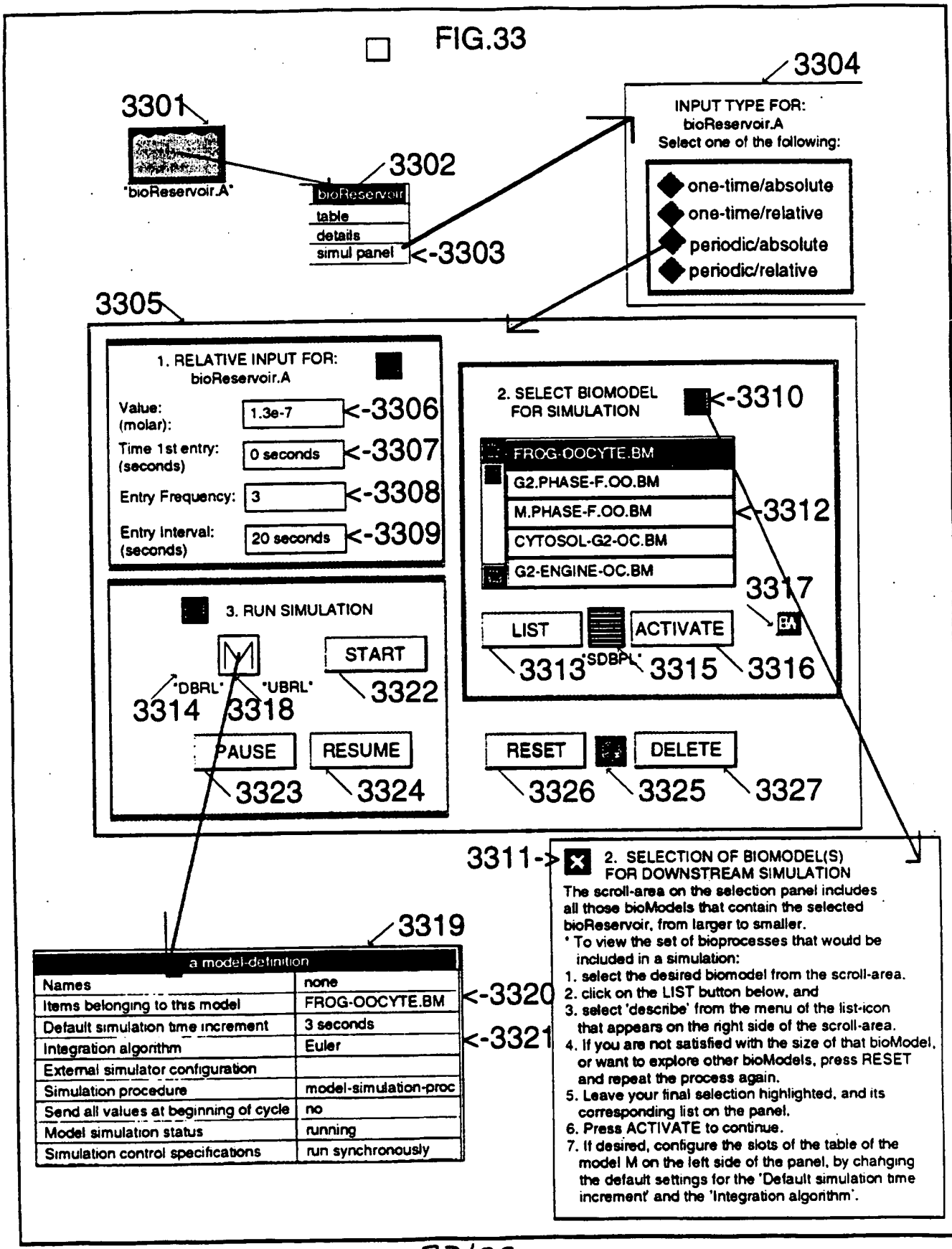

3301
"bioReservoir.A"
3302
bioReservoir
table
details
simul panel
<-3303
3304
INPUT TYPE FOR:
bioReservoir.A
Select one of the following:
one-time/absolute
one-time/relative
periodic/absolute
periodic/relative
3305
1. RELATIVE INPUT FOR:
bioReservoir.A
Value: (molar):
1.3e-7
<-3306
Time 1st entry: (seconds)
0 seconds
<-3307
Entry Frequency:
3
<-3308
Entry Interval: (seconds)
20 seconds
<-3309
2. SELECT BIOMODEL FOR SIMULATION
<-3310
FROG-OOCYTE.BM
G2.PHASE-F.OO.BM
M.PHASE-F.OO.BM
CYTOSOL-G2-OC.BM
G2-ENGINE-OC.BM
<-3312
3317
LIST
"SDBPL"
ACTIVATE
3313
3315
3316
3. RUN SIMULATION
"DBRL"
"UBRL"
3314
3318
START
3322
PAUSE
RESUME
3323
3324
RESET
DELETE
3326
3325
3327
3311->
2. SELECTION OF BIOMODEL(S) FOR DOWNSTREAM SIMULATION
The scroll-area on the selection panel includes all those bioModels that contain the selected bioReservoir, from larger to smaller.
* To view the set of bioprocesses that would be included in a simulation:
1. select the desired biomodel from the scroll-area.
2. click on the LIST button below, and
3. select 'describe' from the menu of the list-icon that appears on the right side of the scroll-area.
4. If you are not satisfied with the size of that bioModel, or want to explore other bioModels, press RESET and repeat the process again.
5. Leave your final selection highlighted, and its corresponding list on the panel.
6. Press ACTIVATE to continue.
7. If desired, configure the slots of the table of the model M on the left side of the panel, by changing the default settings for the 'Default simulation time increment' and the 'Integration algorithm'.
3319
a model-definition
Names
none
Items belonging to this model
FROG-OOCYTE.BM
<-3320
Default simulation time increment
3 seconds
Integration algorithm
Euler
<-3321
External simulator configuration
Simulation procedure
model-simulation-proc
Send all values at beginning of cycle
no
Model simulation status
running
Simulation control specifications
run synchronously

FIG.34

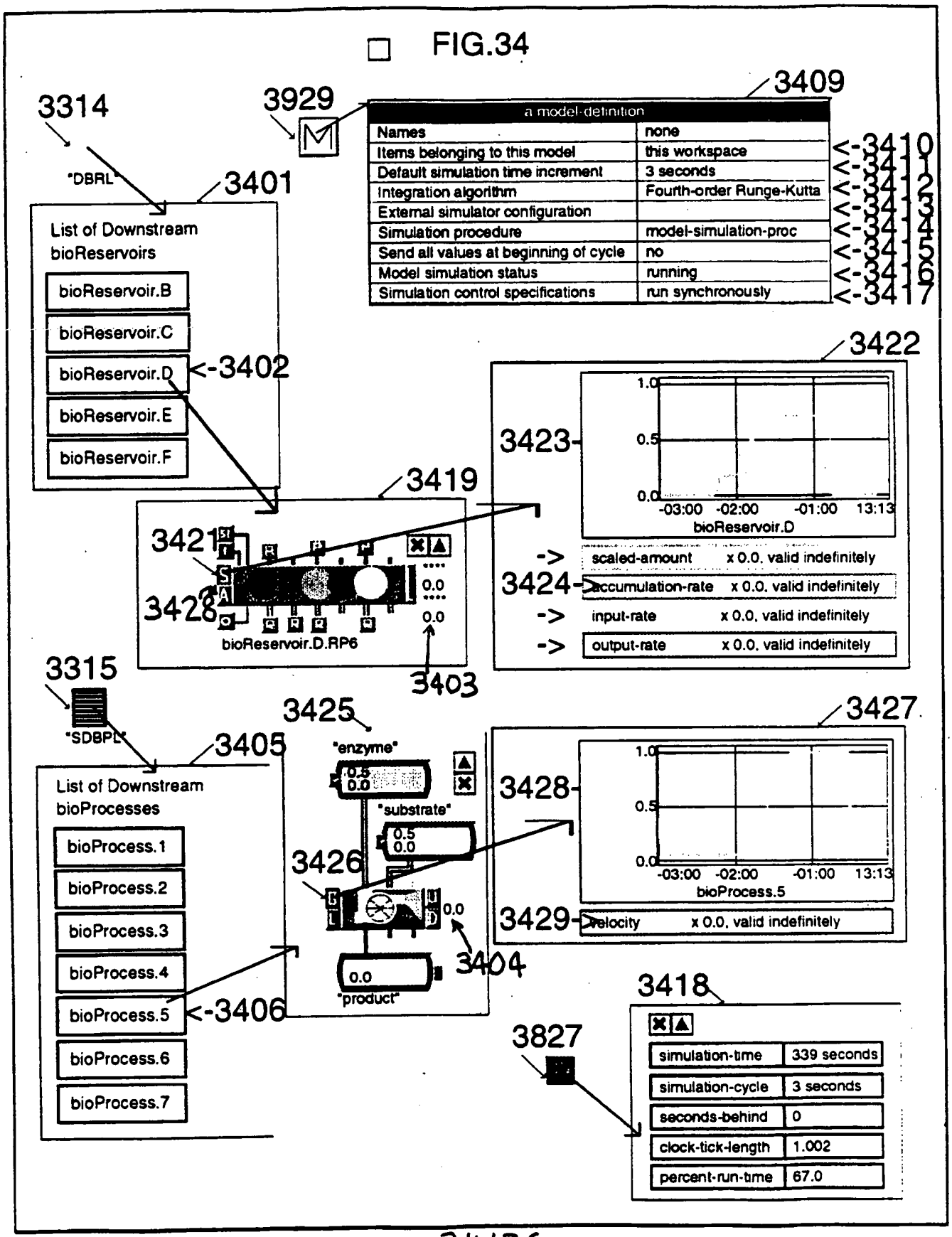

3314
3929
3409
a model-definition
Names
none
Items belonging to this model
this workspace
Default simulation time increment
3 seconds
Integration algorithm
Fourth-order Runge-Kutta
External simulator configuration
Simulation procedure
model-simulation-proc
Send all values at beginning of cycle
no
Model simulation status
running
Simulation control specifications
run synchronously
<-3410
<-3411
<-3412
<-3413
<-3414
<-3415
<-3416
<-3417
"DBRL"
3401
List of Downstream bioReservoirs
bioReservoir.B
bioReservoir.C
bioReservoir.D
bioReservoir.E
bioReservoir.F
<-3402
3422
3423
1.0
0.5
0.0
-03:00 -02:00 -01:00 13:13
bioReservoir.D
3419
3421
3428
0.0
0.0
bioReservoir.D.RP6
-> scaled-amount x 0.0, valid indefinitely
3424
-> accumulation-rate x 0.0, valid indefinitely
-> input-rate x 0.0, valid indefinitely
-> output-rate x 0.0, valid indefinitely
3403
3315
"SDBPL"
3405
List of Downstream bioProcesses
bioProcess.1
bioProcess.2
bioProcess.3
bioProcess.4
bioProcess.5
bioProcess.6
bioProcess.7
<-3406
3425
"enzyme"
0.5
0.0
"substrate"
0.5
0.0
3426
0.0
0.0
"product"
3404
3427
3428
1.0
0.5
0.0
-03:00 -02:00 -01:00 13:13
bioProcess.5
3429
velocity x 0.0, valid indefinitely
3418
3827
simulation-time 339 seconds
simulation-cycle 3 seconds
seconds-behind 0
clock-tick-length 1.002
percent-run-time 67.0

FIG.35

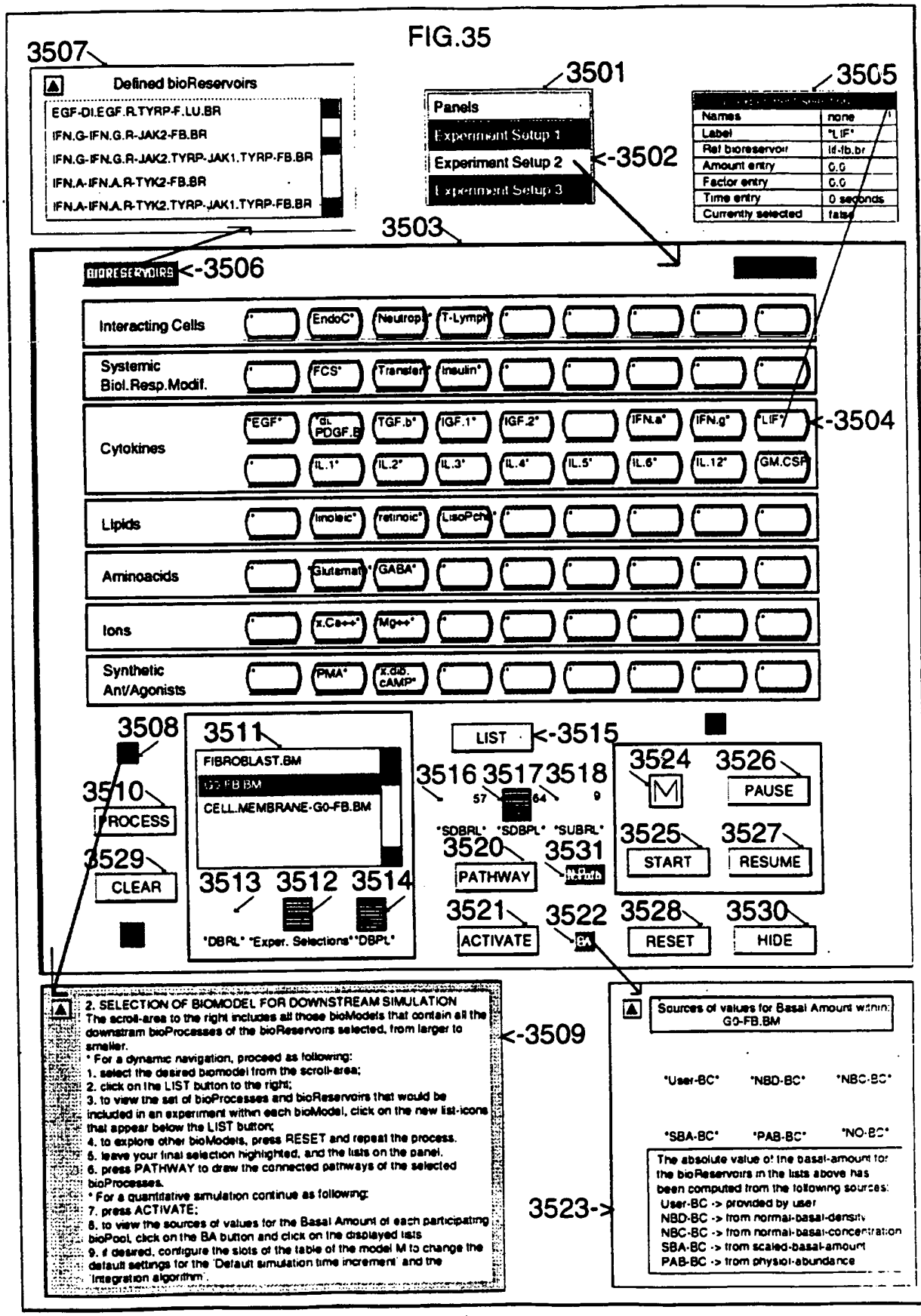
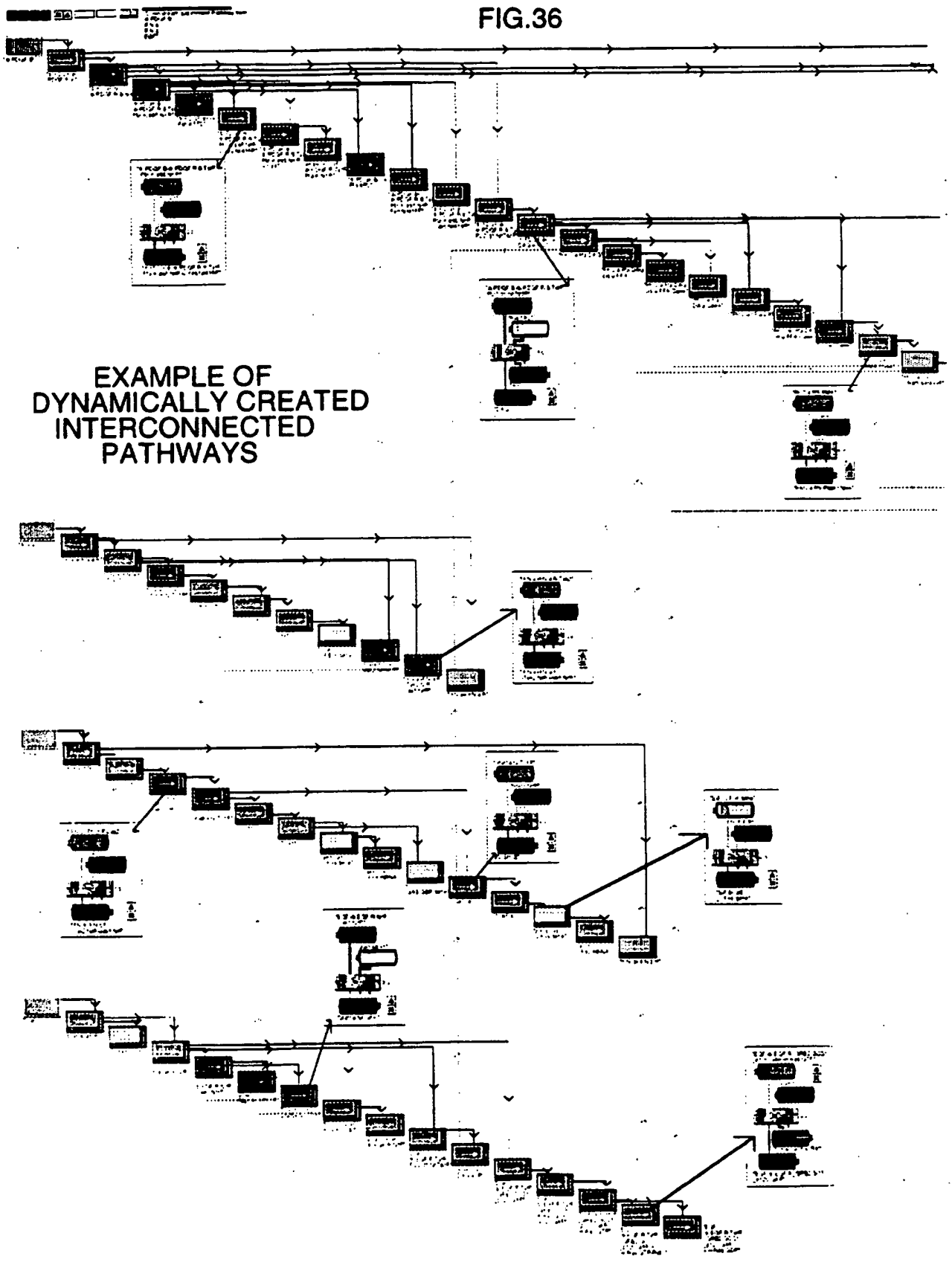

3507
Defined bioReservoirs
EGF-DI.EGF.R.TYRP-F.LU.BR
IFN.G-IFN.G.R-JAK2-FB.BR
IFN.G-IFN.G.R-JAK2.TYRP-JAK1.TYRP-FB.BR
IFN.A-IFN.A.R-TYK2-FB.BR
IFN.A-IFN.A.R-TYK2.TYRP-JAK1.TYRP-FB.BR
3501
Panels
Experiment Setup 1
Experiment Setup 2
Experiment Setup 3
3502
3505
Names
none
Label
"LIF"
Ref bioreservoir
Id-fb.br
Amount entry
0.0
Factor entry
0.0
Time entry
0 seconds
Currently selected
false
3503
BIORESERVOIRS
3506
Interacting Cells
EndoC°
Neutrop°
T-Lymph°
Systemic Biol.Resp.Modif.
FCS°
Transfer°
Insulin°
Cytokines
°EGF°
°PDGF.B°
TGF.b°
IGF.1°
IGF.2°
IFN.a°
IFN.g°
LIF°
3504
IL.1°
IL.2°
IL.3°
IL.4°
IL.5°
IL.6°
IL.12°
GM.CSF°
Lipids
linoleic°
retinoic°
LisoPch°
Aminoacids
Glutamat°
GABA°
Ions
x.Ca++°
Mg++°
Synthetic Ant/Agonists
PMA°
x.db. cAMP°
3508
3510
PROCESS
3529
CLEAR
3511
FIBROBLAST.BM
CELL.MEMBRANE-G0-FB.BM
3513
3512
3514
°DBRL° °Exper. Selections° °DBPL°
LIST
3515
3516
3517
3518
57
64
9
°SDBRL° °SDBPL° °SUBRL°
3520
PATHWAY
3531
3521
ACTIVATE
3522
3524
M
3526
PAUSE
3525
START
3527
RESUME
3528
RESET
3530
HIDE
2. SELECTION OF BIOMODEL FOR DOWNSTREAM SIMULATION
The scroll-area to the right includes all those bioModels that contain all the downstream bioProcesses of the bioReservoirs selected, from larger to smaller.
° For a dynamic navigation, proceed as following:
1. select the desired biomodel from the scroll-area;
2. click on the LIST button to the right;
3. to view the set of bioProcesses and bioReservoirs that would be included in an experiment within each bioModel, click on the new list-icons that appear below the LIST button;
4. to explore other bioModels, press RESET and repeat the process.
5. leave your final selection highlighted, and the lists on the panel.
6. press PATHWAY to draw the connected pathways of the selected bioProcesses.
° For a quantitative simulation continue as following:
7. press ACTIVATE;
8. to view the sources of values for the Basal Amount of each participating bioPool, click on the BA button and click on the displayed lists
9. if desired, configure the slots of the table of the model M to change the default settings for the 'Default simulation time increment' and the 'Integration algorithm'.
3509
Sources of values for Basal Amount within: G0-FB.BM
°User-BC°
°NBD-BC°
°NBC-BC°
°SBA-BC°
°PAB-BC°
°NO-BC°
The absolute value of the basal-amount for the bioReservoirs in the lists above has been computed from the following sources:
User-BC -> provided by user
NBD-BC -> from normal-basal-density
NBC-BC -> from normal-basal-concentration
SBA-BC -> from scaled-basal-amount
PAB-BC -> from physiol-abundance
3523

FIG.36



INTERNATIONAL SEARCH REPORT

'ernational Application No
PCT/US 96/00883

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F19/00 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 367 544 (INT DEV RES CENTER) 9 May 1990 see page 4, line 41 - page 8, line 34 ---	1
X	ELEKTRONIK, vol. 40, no. 22, 29 October 1991, pages 122-123, 128 - 134, XP000268407 KIEZULAS C J ET AL: "ENTWICKLUNG VON EXPERTENSYSTEMEN" see page 128, left-hand column, line 10 - page 133, right-hand column, line 3; figures 4,8 -----	1

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 June 1996	04.07.96

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+ 31-70) 340-3016	Messelken, M

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

ernational Application No
PCT/US 96/00883

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-367544	09-05-90	JP-A- 2244088	28-09-90